Castaneda *et al. J Big Data* (2019) 6:72
https://doi.org/10.1186/s40537-019-0233-0

Journal of Big Data

RESEARCH Open Access

Evaluation of maxout activations in deep learning across several big data domains

Gabriel Castaneda*, Paul Morris and Taghi M. Khoshgoftaar

*Correspondence:
gcastaneda2012@fau.edu
Florida Atlantic University,
Boca Raton, USA

Abstract

This study investigates the effectiveness of multiple maxout activation function variants on 18 datasets using Convolutional Neural Networks. A network with maxout activation has a higher number of trainable parameters compared to networks with traditional activation functions. However, it is not clear if the activation function itself or the increase in the number of trainable parameters is responsible in yielding the best performance for different entity recognition tasks. This paper investigates if an increase in the number of convolutional filters on traditional activation functions performs equal-to or better-than maxout networks. Our experiments compare the Rectified Linear Unit, Leaky Rectified Linear Unit, Scaled Exponential Linear Unit, and Hyperbolic Tangent activations to four maxout function variants. We observe that maxout networks train relatively slower than networks with traditional activation functions, e.g. Rectified Linear Unit. In addition, we found that on average, across all datasets, the Rectified Linear Unit activation function performs better than any maxout activation when the number of convolutional filters is increased. Furthermore, adding more filters enhances the classification accuracy of the Rectified Linear Unit networks, without adversely affecting their advantage over maxout activations with respect to network-training speed.

Keywords: Maxout networks, Activation functions, Big data, Deep learning

Introduction

Deep networks have become very useful for many computer vision applications. Deep neural networks (DNNs) are models composed of multiple layers that transform input data to outputs while learning increasingly higher-level features. Deep learning relies on learning several levels of hierarchical representations for data. Due to their hierarchical structure, the parameters of a DNN can generally be tuned to approximate target functions more effectively than parameters in a shallow model [1]. Today, the typical number of network layers used in deep learning range from five to more than a thousand [2].

Activation functions are used in neural networks (NN) to transform the weighted sum of input and biases, of which is used to decide if a neuron can be fired or not [3]. Commonly used activation functions (nonlinearities) include sigmoid, Hyperbolic Tangent (tanh) and Rectified Linear Unit (ReLU) [4]. The use of ReLU was a breakthrough that enabled the fully supervised training of state-of-the-art DNNs [5]. Compared to traditional activation functions, like the logistic sigmoid units or tanh units, which are anti-symmetric, ReLU is one-sided. This property encourages the hidden units to be sparse,

and thus more biologically plausible [6]. Because of its simplicity and effectiveness, ReLU became the default activation function used across the deep learning community [7]. A Convolutional Neural Network (CNN) using ReLU as its activation function classified 1.2 million images of the ImageNet dataset into 100 classes with an error rate of 37.5% [5]. The deep network implemented by Severyn and Moschitti [8] using ReLU as the activation function demonstrated state-of-the-art performance at both the phrase-level and message-level for Twitter sentiment analysis. At Semeval-2015 (International Workshop on Semantic Evaluation), Severyn and Moschitti's models ranked first in the phrase-level subtask A and second in the message-level subtask B.

The ReLU function saturates when inputs are negative. These saturation regions cause gradient diffusion and block gradients from propagating to deeper layers [9]. Furthermore, ReLUs can die out during learning, consequently blocking error gradients and learning nothing [10]. For these reasons, different activation functions have been proposed for DNN training. There is a lack of consensus on how to select a good activation function for a DNN, and a specific function may not be suitable for all applications. Since an activation function is generally applied to the outputs of all neurons, its computational complexity will contribute heavily to the overall execution time [11]. Most research works on the activation functions are focused on the complexity of the nonlinearity that an activation function can provide [12], or how fast it can be executed [13], but often neglect the impact on different classification tasks.

The maxout nonlinearity [14] selects the maximum value within a group of different outputs (feature maps) and is usually combined with dropout [15], which is widely used to regularize deep networks to prevent overfitting. In NNs, the maxout activation takes the maximum value of the pre-activations. Figure 1 shows two pre-activations per maxout unit, each of these pre-activations has a different set of weights from the inputs denoted as "V". Each hidden unit takes the maximum value over the j units of a group: $h_i = max_j\ Z_{ij}$ where Z is the lineal pre-activation value, i is the number of maxout units and j the number of pre-activation values. Maxout chooses the maximum of n input features to produce each output feature in a network, the simplest

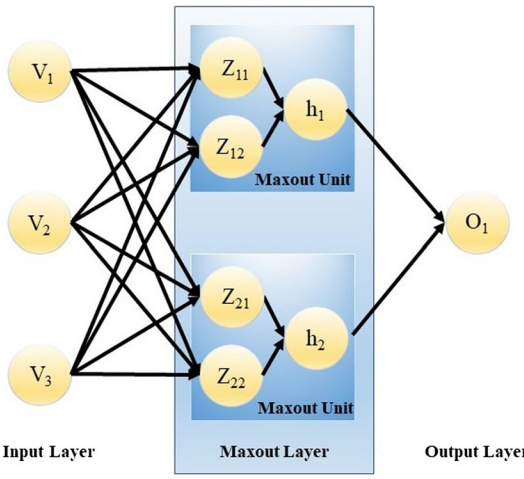

Fig. 1 Two maxout units

case of maxout is the Max-Feature-Map (MFM) [16], where $n = 2$. The MFM maxout computes the function $\max(w_1^T x + b_1, w_2^T + b_2)$, and both the ReLU and leaky ReLU are a special case of this form. When specific weight values w_1, b_1, w_2 and b_2 of the MFM inputs are learned, MFM can emulate ReLU and other rectified linear variants. The maxout unit is helpful for tackling the problem of vanishing gradients because the gradient can flow through every maxout unit [17].

Figure 2 shows the maxout unit in a CNN architecture, where x is a 10×10 pixel image. The maxout unit takes the maximum value of the convolution operations *y1* and *y2*. The CNN learns the weights and bias in the filters *F1* and *F2*. Dropout randomly drops units or connections to prevent units from overfitting. It has been shown to improve classification accuracy in some computer vision tasks [18]. Park and Kwak [19] observed that dropout in CNNs regularizes the networks by adding noise to the output feature maps of each layer, yielding robustness to variations of images. In 2015, the Maxout network In Network (MIN) [17] method achieves in 2015 state-of-the-art or comparable performance on the Mixed National Institute of Standards and Technology (MNIST) [20], the Canadian Institute for Advanced Research (CIFAR-10), CIFAR-100 [21], and Street View House Numbers (SVHN) [22] datasets. Maxout layers were also applied in sentiment analysis [23], with a hybrid architecture consisting of a recurrent neural network stacked on top of a CNN. This approach outperforms a standard convolutional deep neural architecture as well as a recurrent network architecture and performs competitively compared to other methods on two datasets of annotated customer reviews.

CNNs were originally intended for computer vision tasks, being inspired by connections in the visual cortex; however, they have been successfully applied to several DNN acoustic models [24–26] and natural language processing tasks [27, 28]. CNNs are designed to process input features which show local spatial correlations. They can also handle the local translational variance of their input, which makes the network more tolerant to slight position shifts [29].

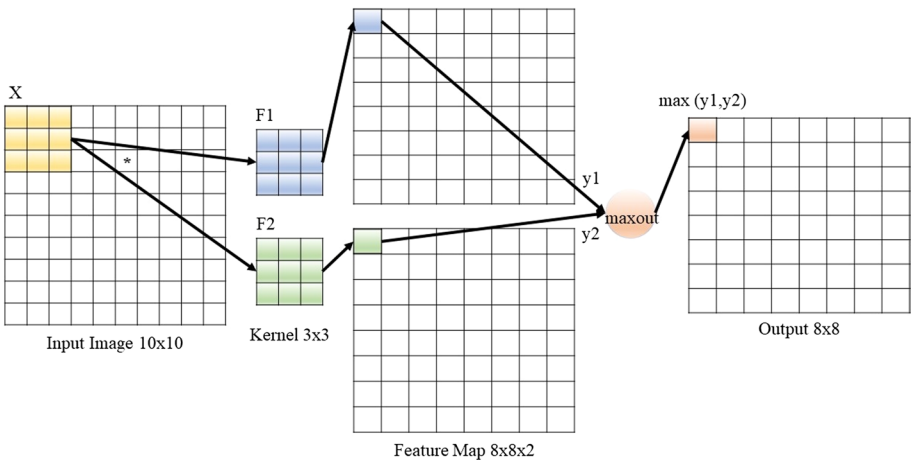

Fig. 2 A CNN maxout unit

The benefit of frequency domain convolution with acoustic models is that CNNs' tolerance to input translations is useful for modeling acoustic data because acoustic models which use convolutions on the frequency domain are robust to speaker and speaking style variations. This is confirmed by studies that experimented with frequency domain convolution. CNNs consistently outperform fully-connected DNNs on the same task [30–32]. When CNNs are used for sentiment analysis, the first layer of the network converts words in sentences to word vectors by table lookup. The word vectors are either trained as part of CNN training, or fixed to those learned by some other method (e.g., word2vec [33]) from an additional large corpus [34]. When working with sequences of words, convolutions allow the extraction of local features around each word.

Most of the comparisons between maxout and other activation functions only report a single performance metric, ignore network size, and only report accuracy on a single dataset, with no training time or memory use analysis. Furthermore, when compared with other activation functions, it is unclear whether marginal performance gains with maxout are due to the activation function or an increase in the number of required trainable parameters. In this work, we evaluated multiple activation functions applied to multiple domains:

- Visual pattern recognition
- Facial verification
- Facial recognition
- Sentiment analysis
- Medical fraud detection
- Sound recognition
- Speech commands recognition.

To the best of our knowledge, this is the first study to evaluate multiple maxout variants and standard activations for multiple domains with significance testing. The main contributions herein can be summarized as follows:

- Evaluate four maxout functions and compare them to popular activation functions like tanh, ReLU, LReLU and SeLU.
- Compare training times for various activation functions.
- Evaluate whether marginal performance gains with maxout are due to the activation function or simply an increase in the number of trainable parameters versus ReLU networks.
- Determine whether maxout methods converge faster and if there is a significant accuracy performance difference between these methods and the standard activations.

The remainder of this paper is organized as follows. The "Related work" section presents related work on activation function evaluation on multiple classification domains. The "Materials and methods" section introduces the activation functions, datasets, and the experimental methodology employed in our experiments. Results

and analysis are provided in "Experimental results and discussion" section. Conclusions with some directions for future work are provided in the "Conclusion" section.

Related work

Maxout is employed as part of deep learning architectures and has been tested against the MNIST, CIFAR-10 and CIFAR-100 benchmark datasets, but it has not been compared against other activation functions using the same deep network architecture and hyperparameters. It is not clear if maxout enhances the overall accuracy on the tested datasets, or if any other activation function has the same effect. There are a few comparisons between maxout and traditional activation functions. Most of the comparisons do not report the details of their network to indicate whether an increased number of filters was accounted for in the experiment.

Most prior work focuses on proposing new activation functions, but few studies have compared different activation functions. Xu et al. [35] investigated the performance of ReLU, leaky ReLU [36], parametric ReLU [37], and the proposed randomized leaky ReLU (RReLU) on three small datasets. In RReLU, the slopes of negative parts are randomized in a given range in the training, and then fixed in the testing. The original ReLU was outperformed by three types of modified leaky ReLU. Mishkin et al. [38] evaluated the influence of activation functions (including ReLU, maxout, and tanh), pooling variants, network width, classifier design, image pre-processing, and learning parameters on the ImageNet dataset. The experiments confirmed the Swietojanski et al. [39] hypothesis about maxout's power in the final layers, as it showed the best performance among non-linear activation functions with speed close to ReLU. The bounded ReLU (brelu), bounded leaky ReLU (blrelu), and bounded bi-firing (bbifire) were presented in [11], and evaluated on classification of basic handwritten digits in MNIST database, complex handwritten digits from the mnist-rot-bg-img database, and facial recognition using the AR Purdue database. Experimental results for all three datasets demonstrate the superiority of all the proposed activation functions, with significant accuracy improvements up to 17.31%, 9.19%, and 74.99% on MNIST, mnist-rot-bg-img, and AR Purdue databases respectively. In [7], automated search techniques were used to discover novel activation functions. The activation function that tends to work better than ReLU on deeper models across three datasets was $h(x) = x \cdot sigmoid(\beta x)$ named Swish, where β is either a constant or a trainable parameter. Only scalar activation functions were used in this study, this limitation would not allow the authors to find or evaluate the maxout activation.

Chang and Chen [17] presented the MIN network. It recorded 0.24%, 6.75% and 28.86% error rates on the MNIST, CIFAR-10, and CIFAR-100, respectively. These error rates are the lowest compared to Network in Network (NIN) [40] and other NIN based networks such as Maxout Network [14] or the Maxout Network in Maxout Network (MIM) [41]. Oyedotun et al. [42] proposed a deep network with maxout units and elastic net regularization. On the MNIST dataset, it reached an error rate of 0.36% and 2.19% on the USPS dataset, surpassing the human performance error rate of 2.5% and all previously reported results. In [43], the Rectified Hyperbolic Secant (ReSech) activation function was proposed and evaluated on MNIST, CIFAR-10, CIFAR-100, and the Pang and Lee's movie review datasets. The results suggest that ReSech units are expected to

produce similar or better results compared to ReLU units for various sentiment prediction tasks. The maxout network accuracy was only compared with the CIFAR-10 and MNIST datasets. Goodfellow et al. [44] investigated the catastrophic forgetting problem, testing four activation functions, including maxout, on MNIST and Amazon using two hidden layers followed by a softmax classification layer. The catastrophic forgetting problem is when a machine learning model is trained on one task, and when trained on a second task it forgets how to perform the first task. Their experiments showed that training with dropout is beneficial, at least on the relatively small datasets used in the paper. Also, the choice of activation function should always be cross-validated, if computationally feasible. Maxout in combination with dropout showed the lowest test errors on all experiments.

The maxout activation is effective in speech recognition tasks [45], but it has not been widely tested on sentiment analysis. Jebbara and Cimiano [23] used the maxout activation in their CNN portion of a hybrid architecture consisting of a recurrent NN stacked on top of a CNN. A maxout layer was also implemented in the Siamese bidirectional Long Short-Term Memory (LSTM) network proposed by Baziotis et al. [46]. The maxout layer was selected as it amplifies the effects of dropout. The output of the maxout layer is connected to a softmax layer which outputs a probability distribution over all classes.

Phoneme recognition tests on the Texas Instruments Massachusetts Institute of Technology (TIMIT) database show that switching to maxout units from rectifier units decreases the error rate for each network configuration studied and yields relative error rate reductions of between 2 and 6% [24]. Zhang et al. [45] introduced two new types of generalized maxout units the p-norm and soft-maxout. In experiments on the Large Vocabulary Continuous Speech Recognition (LVCSR) tasks in various languages, the p-norm units perform consistently better than various versions of maxout, tanh and ReLU. In addition, Swietojanski et al. [39] investigated maxout networks for speech recognition. Through the experiments on voice search and short message dictation datasets, it was found that maxout networks converged around three times faster to train and offer lower or comparable word error rates on several tasks when compared to the networks with logistic nonlinearity. Zhang et al. [47] presented a CNN-based end-to-end speech recognition framework. The maxout unit recorded the lowest error rate compared to ReLU and parametric ReLU.

Using the Public Use File (PUF) data from CMS, Branting et al. [48] proposed graph analysis as a framework for healthcare fraud risk assessment. Their algorithm was evaluated on the Part B (2012–2014), Part D (2013) and List of Excluded Individuals/Entities (LEIE) datasets. Using tenfold cross-validation on the full 12,000-member and 11-feature dataset, the mean f-measure was 0.919 and the mean Receiver Operating Characteristic (ROC) area was 0.960. Sadiq et al. [49] use the 2014 CMS Part B, Part D, and DMEPOS datasets (using only the provider claims from Florida) in order to find anomalies that possibly point to fraudulent or anomalous behavior. A novel framework based on Patient Rule Induction Method (PRIM) was presented, where abnormal behaviors of the physicians are detected. The experimental results show that their framework can effectively shrink the target dataset and deduce a potential suspect subset of physicians who submit several anomalous claims and probably qualify as fraudsters. The attribute sub-space and their correlations are used in PRIM to characterize the low conditional

probability region. The attribute space was characterized by PRIM, which provides a deeper understanding of how certain attributes are the key predictors in identifying fraud. Herland et al. [50] focused on the detection of Medicare fraud using the CMS Part B, Part D, and DMEPOS datasets. A fourth dataset was created by combining the three primary datasets. Based on the area under the ROC curve performance metric, their results show that the combined dataset with the Logistic Regression (LR) learner yielded the best overall score at 0.816, closely followed by the Part B dataset with LR at 0.805.

Our study evaluates 11 activation functions using deep CNN and NN architectures. As opposed to the papers cited in this section, we evaluate if an increase in the number of filters in ReLU enhances the overall accuracy with significance testing. Furthermore, we compare the training and convergence time for all the evaluated activation functions.

Materials and methods

In this section, we introduce the activation functions, datasets, and the empirical methodology employed in this study. In "Activation functions" section, we introduce each evaluated activation function. In "Datasets" section, we describe the datasets employed in our experiments. In the last "Empirical methodology" section, we present our empirical methodology.

Activation functions

Hyperbolic tangent

A hyperbolic tangent (tanh) function is a ratio between hyperbolic sine and cosine functions of x (Fig. 3):

$$h(x) = \tanh = \frac{\sinh(x)}{\cosh(x)} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{1}$$

Rectified units

Rectified linear unit (ReLU) [4] is defined as:

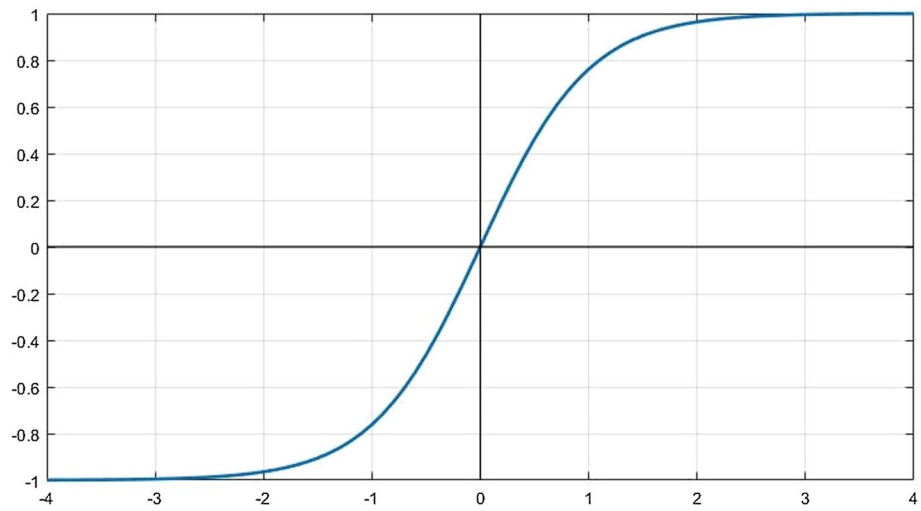

Fig. 3 Hyperbolic tangent

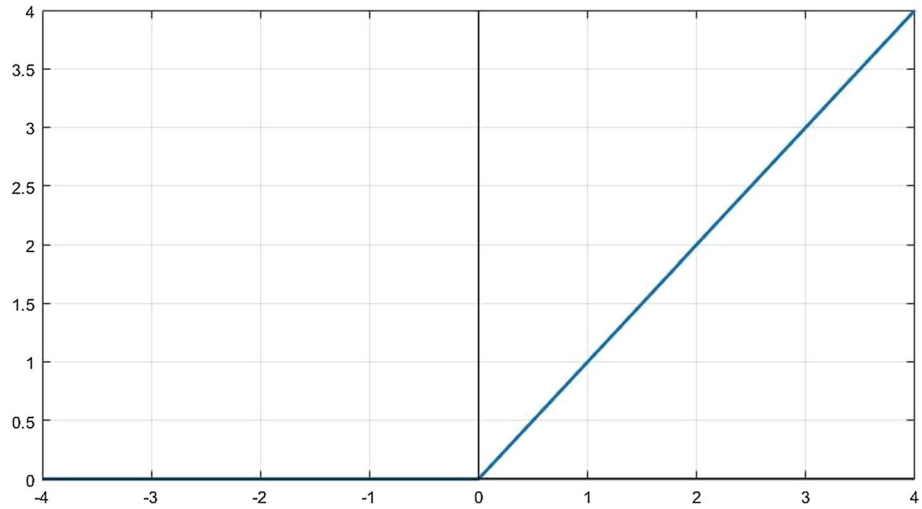

Fig. 4 Rectified linear unit

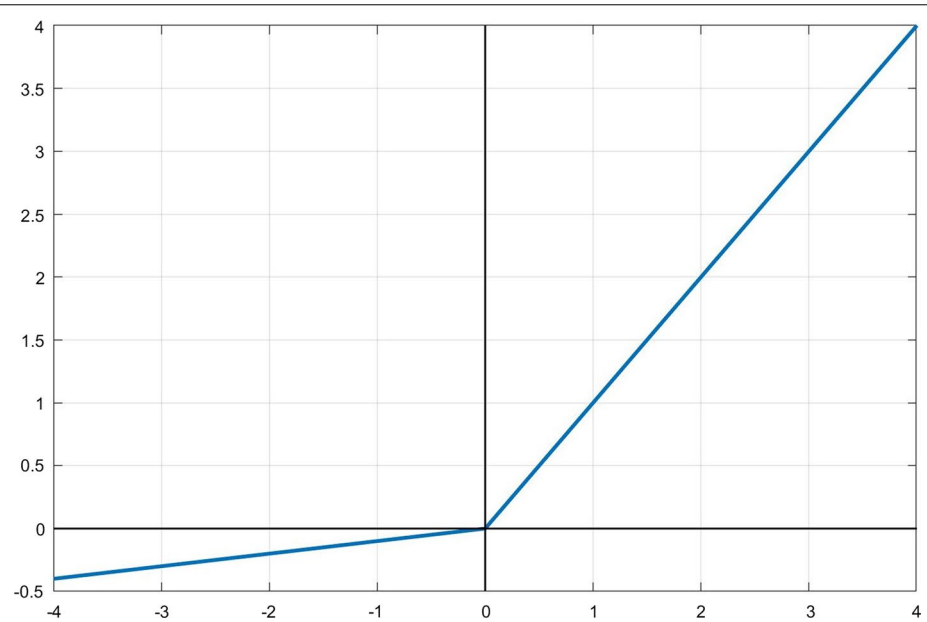

Fig. 5 Leaky rectified linear unit (α = 0.1)

$$h(x) = \max(0, x) \tag{2}$$

where x is the input and $h(x)$ is the output. The ReLU activation is the identity for positive arguments and zero otherwise (Fig. 4).

Leaky ReLU (LReLU) [36] assigns a slope to its negative input. It is defined as:

$$h(x) = \min(0, ax) + \max(0, x) \tag{3}$$

where $a \in (0, 1)$ is a predefined slope (Fig. 5).

The scaled exponential linear unit (SeLU) [51] is given by:

$$h(x) = \lambda \begin{cases} x & if\ x > 0 \\ \alpha e^x - \alpha & if\ x \leq 0 \end{cases} \tag{4}$$

where x is used to indicate the input to the activation function. Klambauer et al. [51] justify why α and λ must have the following values:

$$\begin{aligned} \propto &= 1.673263242354377284817042991671 7 \\ \lambda &= 1.050700987355480493419334985294 6 \end{aligned} \tag{5}$$

to ensure that the neuron activations converge automatically toward an average of 0 and a variance of 1 (Fig. 6).

Maxout units

The maxout unit takes as the input the output of multiple linear functions and returns the largest:

$$\begin{aligned} h(x_i) &= \max w^k \cdot x_i + b^k \\ & k \in \{1, \ldots, K\} \end{aligned} \tag{6}$$

In theory, maxout can approximate any convex function [14], but a large number of extra parameters introduced by the k linear functions of each hidden maxout unit result in large RAM storage memory cost and considerable increase in training time, which affect the training efficiency of very deep CNNs. For our comparisons, we use four variants of the maxout activation: an activation with $k=2$ input neurons for every output (maxout 2-1), an activation with $k=3$ input neurons for every output (maxout 3-1), an activation with $k=6$ input neurons for every output (maxout 6-1), and a variant of maxout with $k=3$ where the two maximum neurons are selected (maxout 3-2). These maxout variants have proven to be effective in classification tasks such as image classification [44], facial recognition [16], and speech recognition [18]. The maxout unit in Fig. 7 mimics a quadratic activation function. The blue quadratic

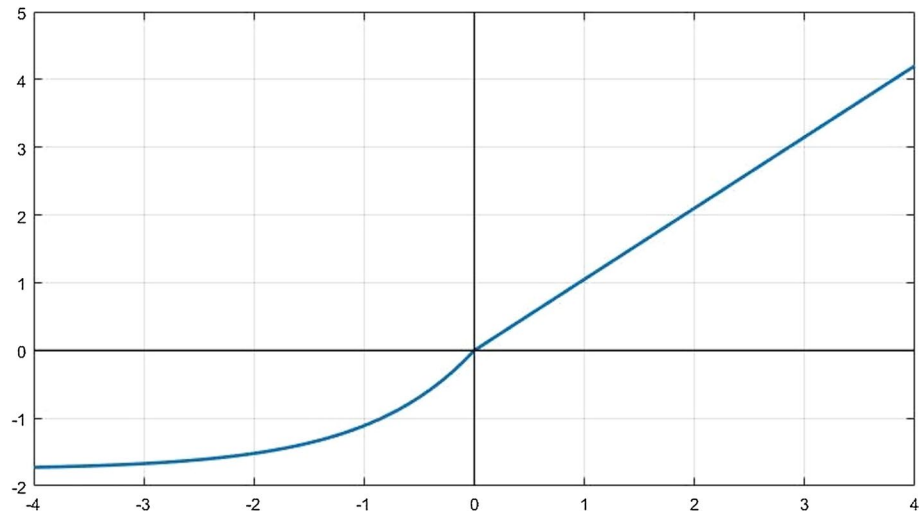

Fig. 6 Scaled exponential linear unit

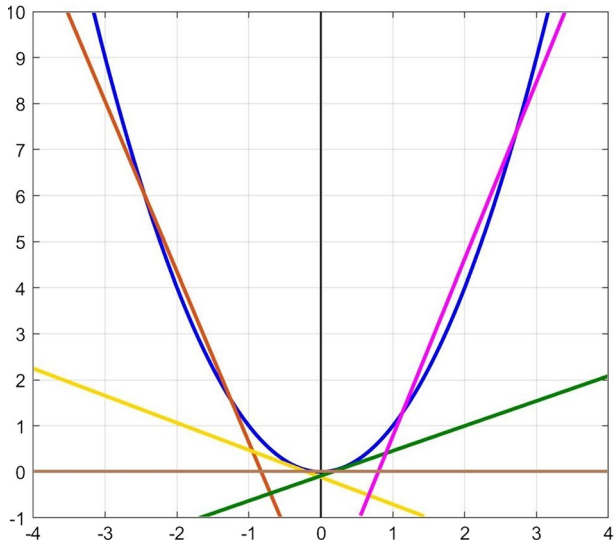

Fig. 7 Maxout (k = 5)

function is not created by the maxout unit, it is only pictured to show what the maxout activation function can approximate when using five linear nodes.

p-norm

The p-norm is the nonlinearity:

$$h = \|x\|_p = \left(\sum_i |x_i|^p \right)^{1/p} \tag{7}$$

where the vector x represents a small group of inputs [45]. If all the x_i were known to be positive, the original maxout would be equivalent to the p-norm with $p = \infty$ (Fig. 8).

Logistic sigmoid

The logistic sigmoid is defined as:

$$h(x) = \frac{1}{1 + e^{-x}} \tag{8}$$

where x is the input. With a range between 0 and 1, the sigmoid function can be used to predict posterior probabilities [52] (Fig. 9).

Datasets

MNIST

The Mixed National Institute of Standards and Technology (MNIST) dataset [20] consists of 8-bit grayscale handwritten digit images, 28 × 28 pixels in size, organized into 10 classes (0 to 9) with 60,000 training and 10,000 test samples.

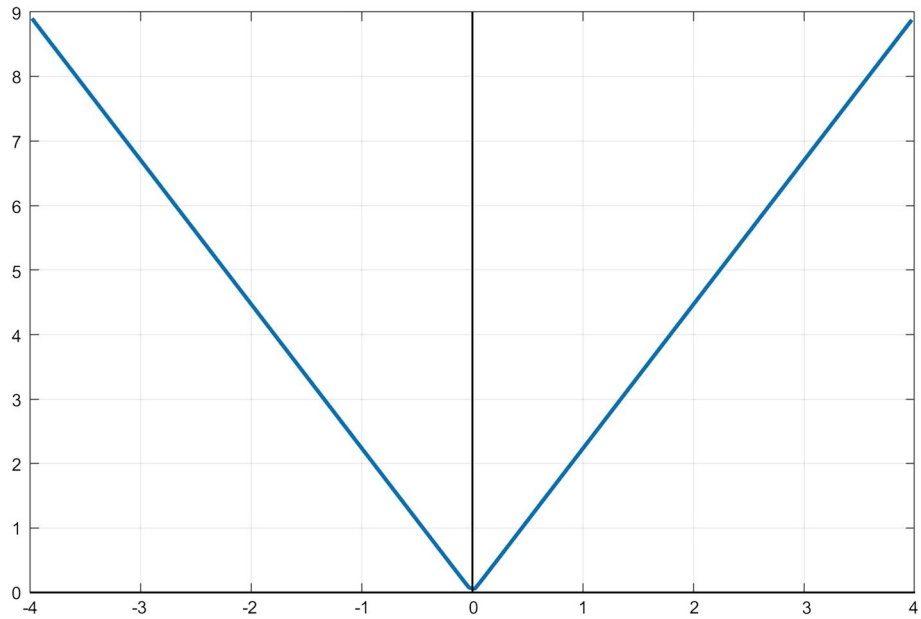

Fig. 8 p-norm (p = 2 i = 5)

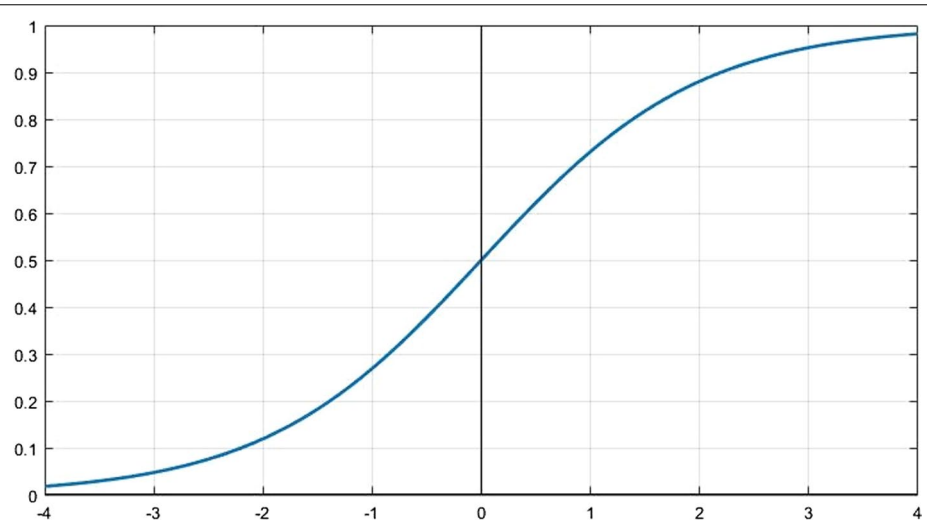

Fig. 9 Logistic sigmoid

Fashion-MNIST

The Fashion-MNIST [53] dataset consists of 60,000 examples where each sample is a 28×28 grayscale image, associated with a label from 10 fashion item classes: T-shirt/top, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, and ankle boot.

CIFAR-10 and 100

The Canadian Institute for Advanced Research (CIFAR)-10 dataset [21] consists of natural color images, 32×32 pixels in size, from 10 classes with 50,000 training and 10,000

test images. The CIFAR-100 dataset is the same size and format as the CIFAR-10; however, it contains 100 classes. Thus, the number of images in each class is only one tenth of that of CIFAR-10.

LFW

The Labeled Faces in the Wild (LFW) dataset contains more than 13,000 images of faces collected from the web by Huang et al. [54] Each face was labeled with the name of the person pictured, with 1680 of the people pictured having two or more distinct photos in the dataset. Images are 250 × 250 pixels in size. The only constraint on these faces is that they were detected by the Viola-Jones face detector [55]. The database was designed for studying the problem of unconstrained facial recognition.

MS-Celeb-1M

The MS-Celeb-1M dataset released by Microsoft [56] contains 10 million celebrity face images for the top 100K celebrities obtained from public search engines, which can be used to train and evaluate both face identification and verification algorithms. There are approximately 100 images for each celebrity, resulting in about 10 million web images. The image resolution is up to 300 × 300 pixels. The authors present a distribution of the million celebrities in different aspects including profession, nationality, age, and gender. The MS-Celeb-1 M is a larger dataset compared to the other test datasets and requires hyperparameter tuning. To avoid the unfair comparison issues associated with changing hyperparameters, we decided to use a manageable subset of 1000 identities which does not require fine-tuning to train. The identities were selected from a cleaned subset of MS-Celeb-1M used for the low-shot learning challenge. These identities were the top 1000 in a list ordered by the number of images.

Amazon product

The original Amazon product review dataset was collected by McAuley et al. [57]. It contains product reviews and scores from 24 product categories sold on Amazon.com, including 142.8 million reviews spanning from May 1996 to July 2014. Review scores lie on an integer scale from 1 to 5. The sentiment dataset constructed from the Amazon product review data in [58] was reused, where 2,000,000 reviews had a score greater than or equal to 4 stars and 2,000,000 reviews had a score less than or equal to 2 stars. The first group is labeled as positive sentiment while the second group is labeled as negative sentiment, creating a positive/negative sentiment dataset. A second subset here called "Amazon1M" was used with one million Amazon product reviews constructed in [59]. The labels were automatically generated from the star rating of each review by assigning a rating below 2.5 as negative and a rating above 2.5 as positive.

Sentiment140

Sentiment140 [60] contains 1.6 million positive and negative tweets, collected and annotated by querying positive and negative emotions, with a tweet considered positive if it contains a positive emoticon like ":)" and negative if, it contains a negative emoticon like ":(".

Yelp

We use the sentiment datasets collected in [59]. It contains 429,061 Yelp reviews from 12 cities in the United States (Yelp500K). This is an imbalanced dataset with 371,292 positive and 57,769 negative instances. Another 500K reviews were scraped to create a second dataset with a million reviews (Yelp1M).

Medicare Part B

The CMS [61] released the Part B dataset [62] and describes Medicare provider claims information for the entire US and its commonwealths, where each instance in the data shows the claims for a provider and procedure performed for a given year. Physicians are identified using their unique National Provider Number (NPI) [63], while procedures are labeled by their Healthcare Common Procedure Coding System (HCPCS) code [64]. Other claims information includes average payments and charges, the number of procedures performed and medical specialty (also known as provider type).

Medicare Part D

The Part D PUF [65] provides information on prescription drugs prescribed by individual physicians and other health care providers and paid for under the Medical Part D Prescription Drug Program. Each physician is denoted by his or her NPI and each drug is labeled by its brand and generic name. Other information includes average payments and charges, variables describing the drug quantity prescribed and medical specialty.

DMEPOS

The Durable Medical Equipment, Prosthetics, Orthotics and Supplies (DMEPOS) PUF [66] presents information on DMEPOS products and services provided to Medicare beneficiaries ordered by physicians and other healthcare professionals. Physicians are identified using their unique NPI within the data while products are labeled by their HCPCS code. Other claims information includes average payments and charges, the number of services/products rented or sold and medical specialty (also known as provider type).

Combined CMS dataset

A combined dataset was created in [50] after processing Part B, Part D, and the DMEPOS datasets, containing all the attributes from each, along with the fraud labels derived from the List of Excluded Individuals and Entities (LEIE). The combining process involves a join operation on NPI, provider type, and year. Due to there not being a gender variable present in the Part D data, the authors did not include this variable in the join operation condition and used the gender labels from Part B while removing the gender labels gathered from the DMEPOS dataset after joining. In combining these datasets, it is limited to those physicians who have participated in all three parts of Medicare.

For each dataset (Part B, Part D and DMEPOS), the information was combined for all available calendar years. For Part B and DMEPOS, all attributes not present in

each available year were removed. The Part D dataset had the same attributes in all available years. For Part B, the standard deviation variables were removed from 2012 to 2013 and standardized payment variables were removed from 2014 to 2015 as they were not available in the other years. For DMEPOS, the standard deviation variable was removed from 2014 to 2015 as it was not available in 2013. For all three datasets, all instances that either were missing both NPI and HCPCS/drug name values or had an invalid NPI were removed. For Part B, all instances with HCPCS codes referring to prescriptions were filtered out. The prescription-related codes are not actual medical procedures, but instead are for specific services listed on the Medicare Part B Drug Average Sales Price file. For the Part B dataset, eight features were kept while the other 22 were removed. For the Part D dataset, seven features were kept and the other 14 were removed. For the DMEPOS dataset nine features were kept and the other 19 were removed. The excluded attributes provide no specific information on the claims, drugs administered, or referrals, but rather encompass provider-related information, such as location and name, as well as redundant variables like text descriptions which can be represented by using the variables containing the procedure or drug codes. For Part D, variables that provided count and payment information for patients 65 or older were not included, as this information is encompassed in the retained variables. The combined dataset contains all the retained features from all three datasets. The purpose of this new dataset is to provide a more encompassing view into a physician's behavior over various branches of Medicare, over individual Medicare parts.

Google speech commands dataset

The Google speech commands (GSC) dataset v0.02 [67] consists of 105,829 one-second long audio files of 35 keywords, by 2618 speakers, with each file consisting of only one keyword encoded as linear 16-bit single-channel PCM values at a 16 kHz rate. A spectrogram using a fast Fourier transform (FFT) is computed for each wave file in the dataset. Frequencies are summed into 129 bins, and each 1-second sample is divided into 71-time bins. The image for each instance is 129×71. The number in each "pixel" represents the power spectral density in dB, and each image is scaled between 0 and 1, relative to the maximum and minimum dB in that image. Samples are not scaled to the maximum and minimum of the whole dataset because recordings were crowdsourced, so the volume for different recordings is not consistent.

IRMAS

The IRMAS dataset [68] is intended to be used for training and testing methods for the automatic recognition of predominant instruments in musical audio. The instruments considered are cello, clarinet, flute, acoustic guitar, electric guitar, organ, piano, saxophone, trumpet, violin, and human singing voice. It includes music from various decades from the past century, hence the difference in audio quality. The training data consists of 6705 audio files with excerpts of 3 s from more than 2000 distinct recordings. The testing data consists of 2874 excerpts with lengths between 5 and 20 s. No tracks from the training data were included. Unlike the training data, the testing data contains one or more predominant target instruments. All audio files are in a 16-bit stereo WAV format sampled at 44.1 kHz. We truncate the recordings in the dataset to 1 s in length and

process them as spectrograms, like the preprocessing of the Google speech commands dataset.

IDMT-SMT-audio-effects

The IDMT-SMT-audio-effects [69] is a dataset for electric guitar and bass audio effects detection. The dataset consists of 55,044 WAV files with a single recorded note of which 20,592 are monophonic bass notes, 20,592 are monophonic guitar notes, and 13,860 are polyphonic guitar sounds. There are 11 different audio effects: feedback delay, slap back delay, reverb, chorus, flanging, phaser, tremolo, vibrato, distortion, overdrive, and no effect (unprocessed notes/sounds). For detailed descriptions of these audio effects please refer to [70].

Empirical methodology

We adopt the general convolutional network architecture demonstrated in recent years to advance the state-of-the-art in supervised classification [21]. We evaluate classification performance on a variety of CNN architectures. In these architectures, a series of convolutional layers for feature extraction are followed by fully-connected layers for classification. Max-pooling is used between convolutional layers to reduce the dimensionality of the network input, and dropout is used before fully-connected layers to prevent overfitting.

A suitable network architecture is selected for each dataset according to the input size and number of instances in the data, as specified in Table 1. Architecture (A) was applied to Medicare Part B, D and combined datasets, architecture (B) to the MNIST, Fashion-MNIST, CIFAR-10, and CIFAR-100 datasets, and architecture (C) to the LFW dataset. The architecture (D) was utilized to the Sentiment140, Yelp, and Amazon datasets, architecture (E) to the Google Speech Commands, IRMAS, and IDMT-SMT-Audio-Effects datasets, and architecture (F) to the MS-Celeb dataset. The depth of the configurations increases from left (A) to right (F), as more layers are added. In general, fewer convolutional layers are used for datasets with smaller number of samples, while deeper architectures are used for larger datasets. Unless otherwise specified, max-pooling is performed with a filter size and stride of 2, and convolutional layer inputs are padded to produce same-size outputs.

Within each dataset, experiments are carried out using the selected CNN architecture, modified only to fit the memory specifications of the activation functions. For example, a CNN for the MNIST dataset with 10 output filters in the first convolutional layer would be modified to output 20 filters as input to a maxout 2-1 activation function. The only layer in each network which is not modified according to the activation function is the final classification layer, where a softmax activation is applied. For networks trained on the MS-Celeb dataset, we use the 9-layer light CNN framework presented in [16], which contains five convolution layers, four NIN layers [40], activation layers and four max-pooling layers. We implement each NIN layer as a convolutional layer with a filter size of one.

The models were trained with a learning rate of 0.01 on all but the MS-Celeb dataset. For this dataset, the learning rate was 0.0001, which is the rate the CNN architecture for the MS-Celeb could converge. Rather than tune each network in our comparison

Table 1 CNN and NN Configurations

Layer	A Medicare Part B Medicare Part D DMEPOS Combined CMS	B MNIST FMNIST CIFAR10 CIFAR100	C LFW	D Sent140 Amazon Yelp	E GSC IRMAS IDMT-SMT-Audio-Effects	F MS-Celeb
Input	123 123 123 123	28 × 28 × 1 28 × 28 × 1 32 × 32 × 3 32 × 32 × 3	128 × 128 × 1	8 × 140 × 1 8 × 500 × 1 8 × 500 × 1	129 × 71 × 1 129 × 71 × 1 100 × 100 × 1	128 × 128 × 1
Conv	N/A	f = 10 k = [5,5] p = none	f = 32 k = [7,7]	f = 64 k = [3,3]	f = 32 k = [5,5]	f = 48 k = [5,5]
Pool	N/A	k = [2,2] s = [2,2]	k = [2,2] s = [2,2]	N/A	k = [2,2] s = [2,2]	k = [2,2] s = [2,2]
C-NIN	N/A	N/A	N/A	N/A	N/A	f = 48 k = [1,1]
Conv	N/A	f = 20 k = [5, 5] p = none	f = 64 k = [5, 5]	f = 64 k = [3,3]	f = 64 k = [3,3]	f = 96 k = [3,3]
Pool	N/A	N/A	k = [2,2] s = [2,2]	N/A	N/A	k = [2,2] s = [2,2]
C-NIN	N/A	N/A	N/A	N/A	N/A	f = 96 k = [1,1]
Conv	N/A	N/A	f = 64 k = [3,3]	f = 64 k = [3,3]	f = 64 k = [3,3]	f = 64 k = [3,3]
Pool	N/A	N/A	N/A	k = [2,2] s = [2,2]	k = [2,2] s = [2,2]	k = [2,2] s = [2,2]
C-NIN	N/A	N/A	N/A	N/A	N/A	f = 192 k = [1,1]
Conv	N/A	N/A	N/A	f = 64 k = [3,3]	f = 64 k = [3,3]	f = 128 k = [3,3]
C-NIN	N/A	N/A	N/A	N/A	N/A	f = 128 k = [1,1]
Conv	N/A	N/A	N/A	f = 64 k = [3,3]	f = 64 k = [3,3]	f = 64 k = [3,3]
Conv	N/A	N/A	N/A	f = 64 k = [3,3]	N/A	N/A
Pool	N/A	N/A	N/A	k = [2,2] s = [2,2]	k = [2,2] s = [2,2]	k = [2,2] s = [2,2]
FC	n = 512	N/A	n = 256	n = 512	n = 512	n = 256
DO	kp = 0.5	kp = 0.5	kp = 0.5	kp = 0.5	kp = 0.5	kp = 0.5
Pool	N/A	k = [2,2] s = [2,2]	k = [2,2] s = [2,2]	N/A	N/A	N/A
FC	n = 64	n = 50 n = 250[a]	N/A	n = 512	n = 512	n = 256
DO	kp = 0.5	kp = 0.5	kp = 0.5	kp = 0.5	kp = 0.5	kp = 0.5
FC	n = 2	n = 10 n = 100[a]	n = 2	n = 2	n = 35 n = 11[b]	n = 1000

Convolutional layers (Conv) indicate the number of filters (f=), the kernel size (k=) and the padding (p=). Convolution network in network layers (C-NIN) indicate the number of filters (f=), and the kernel size (k=). Max-pool layers (Pool) indicate the kernel size (k=) and the stride (s=). Dropout layers (DO) show the applied keep probability (kp=), and the fully-connected layers (FC) display the number of neurons (n=)

[a] Neurons applied only to the CIFAR-100 dataset

[b] Neurons applied only to the IRMAS and IDMT-SMT-Audio-Effects datasets

optimally with a validation set, we implement a set of uniform stopping criteria during training to maintain a consistent protocol so that network performance on a test set is suitable for comparison across activations [21]. Early stopping criteria is the same for every dataset, with the slope of the test loss calculated over a running window of the past three epochs on MNIST, FMNIST, CIFAR and LFW datasets, and past four epochs on the rest of the datasets. When the slope becomes positive, testing loss no longer decreases and network training is stopped. The optimizer is stochastic gradient descent

Table 2 Momentum and batch size per dataset

Dataset	Momentum	Batch size
MNIST	0.5	64
Fashion-MNIST	0.5	64
CIFAR-10	0.5	64
CIFAR-100	0.5	64
LFW	0.5	64
MS-Celeb	0.9	128
Medicare Part B	N/A	200
Medicare Part D	N/A	200
DMEPOS	N/A	200
Combined CMS	N/A	200
Sentiment140	N/A	200
Google Speech Commands	N/A	200
IRMAS	N/A	200
IDMT-SMT-Audio-Effects	N/A	200
Amazon4M	N/A	200
Amazon1M	N/A	200
Yelp500K	N/A	200
Yelp1M	N/A	200

and the loss function is the categorical cross-entropy. Table 2 displays the momentum, and batch size per dataset.

The number of trainable parameters using ReLU and maxout activation functions are presented in Table 3. Doubling the ReLU units not only doubles the number of trainable parameters, each layer will output twice as many feature maps. Unlike the maxout unit with $k=2$ (maxout 2-1) has twice as many parameters, but each unit still outputs one feature map. Similarly, a maxout unit with $k=3$ (maxout 3-1) has 3x the number of parameters, but still only outputs one feature map.

The classification tasks and datasets used to evaluate the activation functions are:

1. Image classification using the MNIST, F-MNIST, CIFAR-10, and CIFAR-100 datasets. These are the most widely used datasets in machine learning, MNIST and CIFAR-10 were the two most common datasets in NIPS 2017 [71].
2. Facial verification using the LFW dataset. The LFW is a popular benchmark dataset that contains diverse illumination conditions combined with variations in pose and expressions. Companies, independent teams and data-scientists use this dataset to verify the quality of their algorithms.
3. Facial recognition using the MS-Celeb dataset. The dataset was tested in [16] using a 9-layer light CNN framework using maxout 2-1. The MS-Celeb-1M dataset contains massive noisy labels, a challenge a facial recognition system has to attenuate and if possible eliminate.
4. Sentiment analysis using two Amazon product data subsets (1 and 4 million reviews), Sentiment140 and two subsets from the Yelp text datasets (500,000 and 1,000,000 reviews). We used the datasets constructed in [59]. This provides us with a dataset consisting of short text (sentiment140) and four datasets with longer instances (Amazon and Yelp reviews).

Table 3 Number of trainable parameters per dataset

Dataset	ReLU	ReLU 2x	ReLU 3x	ReLU 6x	MFM 2-1	MFM 3-1	MFM 6-1	MFM 3-2
Fashion-MNIST	21,840	85,670	191,500	760,990	43,170	64,500	128,490	128,000
MNIST	21,840	85,670	191,500	760,990	43,170	64,500	128,490	128,000
CIFAR-10	31,340	122,670	274,000	1,087,990	62,170	93,000	185,490	183,500
Medicare Part B	96,450	258,434	485,954	1,561,730	192,770	289,090	578,050	387,522
Medicare Part D	97,474	260,482	489,026	1,567,874	194,818	292,162	584,194	390,594
DMEPOS	105,666	276,866	513,602	1,617,026	211,202	316,738	633,346	415,170
Combined CMS	120,002	305,538	556,610	1,703,042	239,874	359,746	719,362	458,178
CIFAR-100	156,130	572,160	1,248,190	4,836,280	287,160	418,190	811,280	833,190
IDMT-SMT-Audio-Effects	825,996	3,294,476	7,405,452	29,593,356	1,648,908	2,471,820	4,940,556	4,938,636
IRMAS	1,808,779	7,226,123	16,252,043	64,981,259	3,614,731	5,420,683	10,838,539	10,836,363
Sentiment140	2,743,234	10,966,914	24,671,042	98,666,114	5,485,442	8,227,650	16,454,274	16,449,346
MS-Celeb	2,997,432	11,469,704	25,417,816	100,117,192	5,737,864	8,478,296	16,699,592	16,948,056
LFW	4,286,434	17,137,602	38,553,506	154,189,634	8,572,354	12,858,274	25,716,034	25,705,890
Google Speech Commands	6,767,267	27,028,771	60,784,547	243,017,507	13,516,579	20,265,891	40,513,827	40,525,219
Amazon1M	8,641,474	34,559,874	77,755,202	311,002,754	17,281,922	25,922,370	51,843,714	51,838,786
Yelp500K	8,641,474	34,559,874	77,755,202	311,002,754	17,281,922	25,922,370	51,843,714	51,838,786
Yelp1M	8,641,474	34,559,874	77,755,202	311,002,754	17,281,922	25,922,370	51,843,714	51,838,786
Amazon4M	17,030,082	68,114,306	153,252,674	612,992,642	34,059,138	51,088,194	102,175,362	102,170,434

5. Fraud detection using the Medicare Part B, Part D, DMEPOS, and combined Medicare datasets. The datasets were preprocessed, and the combined CMS created in [50]. The datasets focus on the detection of Medicare fraud using the Medicare Provider Utilization and Payment Data: Physician and Other Supplier (Part B), Medicare Provider Utilization and Payment Data: Part D Prescriber (Part D), and Medicare Provider Utilization and Payment Data: Referring Durable Medical Equipment, Prosthetics, Orthotics and Supplies (DMEPOS).
6. Speech command classification using the Google speech commands dataset. The dataset contains single-word speaking commands that can be converted to 129×71 pixel images. Other speech command datasets have recordings with a longer duration causing memory constraints when testing maxout activation functions.
7. Audio classification using the IRMAS and IDMT-SMT-Audio-Effects datasets. Similar to the Google speech commands dataset, the IRMAS and IDMT-SMT-Audi-Effects datasets were truncated to 1 s of audio and converted to 129×71 and 100×100 pixel images respectively.

Face verification, or authentication is a one-to-one (1:1) match that compares a test face image against a gallery face image whose identity is being claimed. The current standard for benchmarking performance on unconstrained face verification is the Labeled Faces in the Wild (LFW) dataset [54]. We compare activations on View 2

of the dataset, which consists of image pairs that are labeled either matching or not matching. We process each face image to be grayscale and cropped to 128 × 128 pixels. After preprocessing, we train using 90% of the View 2 pairs and evaluate each network on the final 10% of pairs. Face identification is a one-to-many matching (1:N) process that compares a test face image against all the gallery images in a face database to determine the identity of the test face. Face identification was tested on the MS-Celeb dataset. To facilitate efficient training, we filter the MS-Celeb subset to include slightly over 100,000 face images corresponding to 1000 celebrity classes. We use an additional test set of approximately 10,000 images to validate each network. Each image is cropped to 128 × 128 pixels.

For our sentiment analysis, the text was embedded as proposed by Prusa et al. [72]. The authors propose a log(m) character embedding where each character in the alphabet is given an integer value, where m is the alphabet size. The equivalent binary representation of a character's integer value is then found and turned into a vector of 0s and 1s. This results in a denser representation compared to 1-of-m character embedding [73]. This embedding was tested against the 1-of-m embedding using an alphabet size of 70 and 256. Results show significantly higher performance and a faster training time when using the log(m) representation of the data.

As the Medicare datasets are highly imbalanced, we employ random under-sampling (RUS) to mitigate the adverse effects of class imbalance. RUS is the process of randomly removing instances from the majority class of a dataset in order to balance the ratio (non-fraudulent/fraudulent). We generate a class distribution (majority:minority) of 50:50. There are 2036 samples in Medicare Part D, 2818 in Medicare part B, 1275 in DMEPOS and 946 in the CMS combined dataset.

ReLU is also evaluated with 2x, 3x and 6x the number of filters in each convolutional layer. The purpose of including these variants is to consider the impact of increased neurons on the accuracy, training time and memory usage of NNs independent of the maxout activation. Because maxout incorporates both the max operation and the use of duplicate neurons with additional memory, it is necessary to consider how each component of the activation contributes to its performance.

Maxout is evaluated with the following combinations of input and output neuron quantities: 2-1, 3-1, 3-2 and 6-1. We compute maxout for our four activations using the equations below, which are suitable for parallelization with modern deep learning software and parallel computer hardware. In general, we use maximum (max) and minimum (min) operations with two inputs to achieve maximum computational efficiency during training.

$$maxout\ 2\text{-}1(x_1, x_2) = \max(x_1, x_2) \tag{9}$$

$$maxout\ 3\text{-}1(x_1, x_2, x_3) = \max(x_1, \max(x_2, x_3)) \tag{10}$$

$$maxout\ 6\text{-}1(x_1, x_2, x_3, x_4, x_5, x_6) = \max(x_1, \max(x_2, \max(x_3, \max(x_4, \max(x_5, x_6))))) \tag{11}$$

$$\begin{aligned} maxout\ 3\text{-}2(x_1, x_2, x_3) &= \max(x_1, \max(x_2, x_3)), \\ &\min(\max(x_1, x_2), \min(\max(x_2, x_3), \max(x_1, x_3))) \end{aligned} \tag{12}$$

While it would be ideal to record the wall clock time needed to train each network, modern high-performance computing environments present hardware and software challenges which make it difficult to safely compare training time across runs or activations. Thus, we produce a metric which represents the time cost of training with a particular activation function. This metric is produced for each activation on an isolated desktop computing environment. We record the wall clock time required to train each network in our comparison for 100 batches and take the average time across 10 runs on a single desktop computer with 32 GB of RAM running Ubuntu 16.04 with an intel i7 7th generation CPU and an NVIDIA 1080ti GPU. Those times are produced independently for each activation on each dataset. We note that the 100 batch intervals in our timing measurements do not constitute complete training epochs. Computations are timed over 100 batches to average out any variability in individual batch times and facilitate accurate comparison between activations.

A total of 11 activation functions were evaluated, where each experiment compared:

- Classification accuracy
- Average 100 batches time
- Average 100 batches total training time (average 100 batches time multiplied by number of epochs to converge).

Due to memory constraints, ReLU3x and ReLU6x were not included in comparisons on the text, face classification, fraud detection and audio experiments. These ReLU variants produce layers with 3x and 6x more feature maps than the maxout units causing memory limitations particularly with large datasets. Maxout 6-1 was also excluded from the MS-Celeb task as it was difficult to train without tuning. The training did not converge most of the time with the SeLU activation function on the Amazon and Yelp datasets; out of many runs it was only possible to get one successful training on Amazon1M and Yelp1M. Similarly, maxout 6-1 and maxout 3-2 on Amazon4M failed to converge on large text datasets. In future work, hyperparameter tuning for ReLU3x, ReLU6x, maxout 6-1, and maxout 3-2 will be done to find the best performance for the models. Table 4 displays the number of experiments per activation function and dataset.

In each dataset, we use a train/test split of approximately 90%/10%. The train and test size per dataset are presented in Table 5. Because we apply a consistent early stopping criterion, we report results of our comparison done directly on a test set, without an additional validation set. We implemented our models in Keras [74] with TensorFlow [75] as the backend.

Experimental results and discussion

Based on preliminary observations, it is evident the sigmoid activation does not perform well in CNNs, and the maxout p-norm models are very difficult to train. For these reasons, both activation functions are not included in the results.

In order to test the statistical significance of performances of the type of activation (maxout vs. other activations) across all the datasets, one-way analysis of variance (ANOVA) [76] was performed. In this ANOVA test, the results from 544 evaluations were considered together, and all tests of statistical significance utilized a significance

Table 4 Number of experiments per activation and dataset

Dataset	LR	M21	M31	M32	M61	R	R2X	R3X	R6X	SL	T
CIFAR-10	5	5	5	5	5	5	5	5	5	5	5
CIFAR-100	2	2	2	2	2	2	2	2	2	2	2
F-MNIST	5	5	5	5	5	5	5	5	5	5	5
MNIST	5	5	5	5	5	5	5	5	5	5	5
LFW	2	2	2	2	2	2	2	2	2	2	2
MS-Celeb	2	2	2	2	0	2	2	0	0	2	2
Amazon1M	2	2	2	2	2	2	2	0	0	1	2
Amazon4M	2	2	2	1	1	2	2	0	0	0	2
Sent140	2	2	2	2	2	2	2	0	0	2	2
Yelp500K	2	2	2	2	2	2	2	0	0	0	2
Yelp1M	2	2	2	2	2	2	2	0	0	1	2
Med Part B	5	5	5	5	5	5	5	0	0	5	5
Med Part D	5	5	5	5	5	5	5	0	0	5	5
DMEPOS	5	5	5	5	5	5	5	0	0	5	5
Combined CMS	5	5	5	5	5	5	5	0	0	5	5
GSC	2	2	2	2	2	2	2	0	0	2	2
IRMAS	2	2	2	2	2	2	2	0	0	2	2
IDMT-SMT-Audio-Effects	2	2	2	2	2	2	2	0	0	2	2

LReLU (LR), Maxout 2-1 (M21), Maxout 3-1 (M31), Maxout 3-2 (M32), Maxout 6-1 (M61), ReLU (R), ReLU2x (R2X), ReLU3x (R3X), ReLU6x (R6X), SeLU (SL), Tanh (T)

Table 5 Train and test size per dataset

Dataset	Train	Test
Amazon4M	3,600,000	400,000
Sentiment140	1,440,000	160,000
Amazon	900,000	100,000
Yelp1M	899,611	99,957
Yelp500K	386,154	42,907
Google Speech Commands	94,824	11,005
MS-Celeb	91,683	5000
MNIST	60,000	10,000
Fashion-MNIST	60,000	10,000
CIFAR-10	50,000	10,000
CIFAR-100	50,000	10,000
IDMT-SMT-Audio-Effects	37,065	4119
IRMAS	18,103	2012
LFW	5400	600
Medicare Part B	2618	200
Medicare Part D	1836	200
DMEPOS	1070	200
Combined CMS	846	100

level α of 5%. The factor is significant if the p value is less than 0.05. The ANOVA table is presented in Table 6. As shown the factor is not significant, indicating the activation type does not make a difference in the classification accuracy (p-value is not less than 0.05).

Table 6 One-way ANOVA for type of activation and classification task

Factors	Sum of squares	Percentage of variation (%)	Degrees of freedom	Mean square	F-computed	*p*-value
Activation type	155.6	0.1	1	155.61	0.62	0.42
Error	133,392	99.90	542	246.11		
Total	133,547.6	100	543			

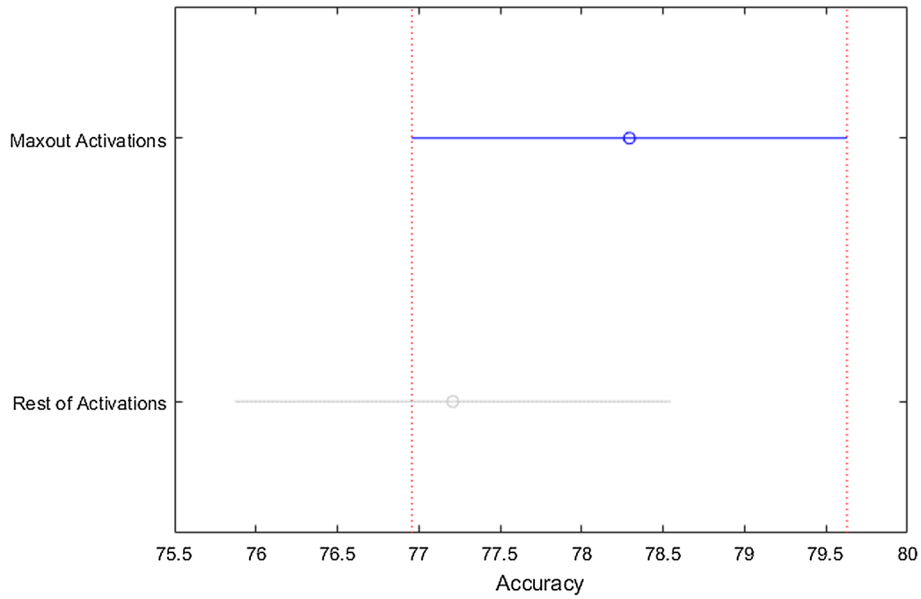

Fig. 10 Multiple comparisons, type of activations

A comparison between maxout and the rest of the activations is presented in Fig. 10, displaying graphs with each group mean represented by the symbol (○) and the 95% confidence interval as a line through the symbol. Maxout activations are not statistically better than the rest of the activation functions.

The best activation average accuracy per dataset, its average time to train 100 batches and average epochs multiplied by average 100 batches time (average 100 batches training time) are presented in Table 7. The activation average accuracy and epochs columns in Table 7 represent the best performing activation function for each dataset. Data in these columns is averaged over the number of training runs specified in Table 4. On the image and face datasets, ReLU with 6x filters reported the highest accuracy on all datasets except the MS-Celeb dataset.

There is no statistical difference between the activation functions, but on average ReLU6x reported the highest accuracy of 85.19% on the image datasets (Fig. 11). The average accuracy presented on the figures, is also the average accuracy over the number of training runs specified in Table 4. Activation function means are significantly different if their intervals are disjoint, and are not significantly different if their intervals overlap. On the facial verification task, ReLU6x also achieved the highest accuracy, while on the facial recognition task, SeLU recorded the highest accuracy (Table 7).

Table 7 Best activation and its results per dataset

Dataset	Best activation	Average accuracy (%)	Average epochs	Average 100 batches time (s)	Average 100 batches training time
CIFAR-10	ReLU6x	79.91	64	0.26	13.33
CIFAR-100	ReLU6x	50.44	60	0.33	18.76
Fashion MNIST	ReLU6x	92.48	89	0.22	17.37
MNIST	ReLU6x	99.46	35	0.22	7.89
LFW	ReLU6x	79.67	51	26.75	1418.12
MS-Celeb	SeLU	97.50	97	39.09	4202.31
All image and face datasets combined	ReLU6x	84.40	60	5.5	161.41
Amazon1M	Maxout 3-2	88.17	35	73.27	2124.86
Amazon4M	Maxout 6-1	93.73	26	57.32	1490.36
Sentiment140	ReLU2x	84.57	60	5.09	259.79
Yelp500K	ReLU2x	93.17	60	18.19	873.33
Yelp1M	ReLU	93.60	60	8.65	519.41
All text datasets combined	ReLU2x	90.41	40	15.57	594.15
Medicare Part B	SeLU	71.0	29	0.12	7.98
Medicare Part D	Maxout 2-1 Maxout 6-1	71.5	180	0.12	22.84
DMEPOS	SeLU	68.5	51	0.12	5.54
Combined CMS dataset	SeLU	74.0	160	0.12	21.05
All Medicare datasets combined	SeLU	69.7	107	0.12	13.01
Google Speech Commands	Maxout 3-2	91.93	45	50.17	2257.65
IRMAS	ReLU2x	67.59	180	10.14	1825.20
IDMT-SMT-Audio-Effects	SeLU	95.51	87	5.94	531.64
All sound datasets combined	Maxout 2-1	83.19	79	11.59	983.18

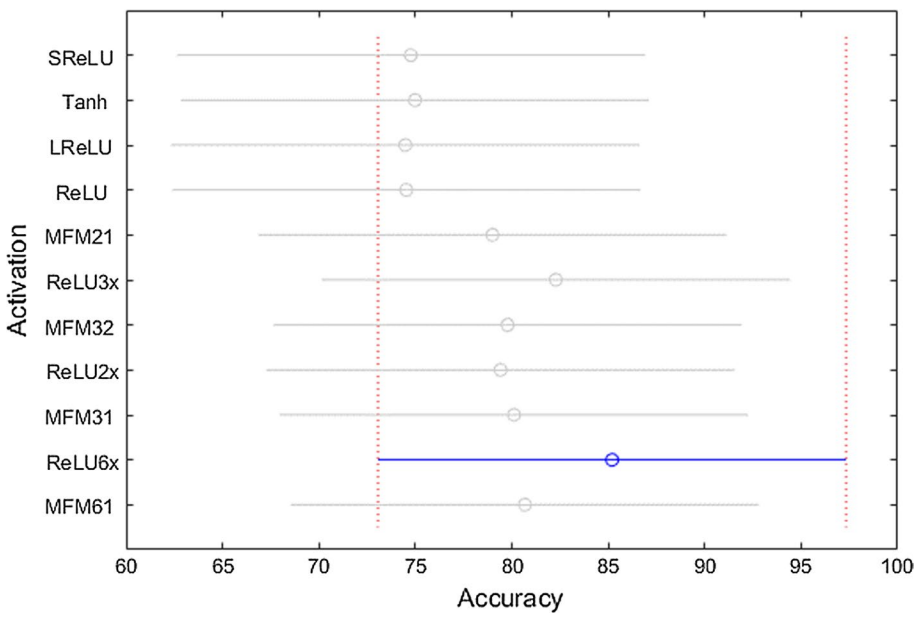

Fig. 11 Multiple comparisons, activations on image datasets

ReLU with 2x filters reported the highest accuracy on Sentiment140 and Yelp500K datasets (Table 7). On Yelp1M, ReLU achieved the highest accuracy and maxout 3-2 and 6-1 reported the best accuracies on the Amazon1M and Amazon4M datasets respectively. On average, ReLU2x reported the highest accuracy of 90.41% (Fig. 12).

SeLU reported the highest accuracy on Medicare Part B, DMEPOS and the combined CMS datasets (Table 7). On Medicare Part D, maxout 2-1 and maxout 6-1 achieved the highest accuracy. On average, SeLU reported the highest accuracy of 69.7% (Fig. 13). This suggests that SeLU is effective for the medical fraud detection task using a NN. On the Medicare Part D dataset, maxout 2-1 and maxout 6-1 obtained the highest accuracy, followed by SeLU, ReLU2x and maxout 3-2 with a 0.5% difference in value. This confirms that SeLU is also effective in this dataset.

Maxout 3-2 reported the highest accuracy on Google speech commands, and on IRMAS ReLU2x achieved the highest accuracy. SeLU recorded the highest accuracy on the IDMT-SMT-Audio-Effects dataset (Table 7). On average, maxout 2-1 reported the highest accuracy of 83.19% (Fig. 14). This indicates maxout 2-1 is effective for the sound recognition task using spectrograms in combination with CNNs.

ReLU with 6x filters delivered the highest accuracy on all image datasets and the LFW dataset. Although ReLU6x and ReLU3x were not tested on the text datasets, other ReLU variants continued to record the highest accuracies. ReLU with 2x filters and ReLU obtained the highest accuracies on the text datasets except for the Amazon datasets (Table 7). A similar result occurred on the sound datasets. Although on average, maxout 2-1 recorded the highest accuracy, ReLU2x reported a 0.36% difference in value. Adding multiple layers of filters was enough for ReLU to achieve the highest classification accuracy on the image and text datasets. This suggests that adding more layers on ReLU2x could increase the accuracy on the MS-Celeb, Amazon, IDMT-SMT-Audio-Effects and Google speech commands datasets, but hyperparameter tuning might be required.

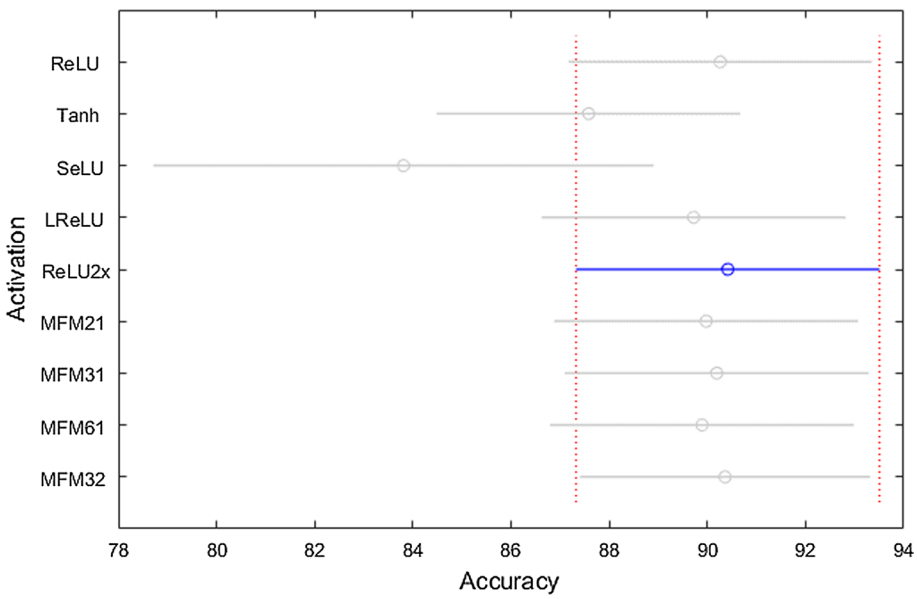

Fig. 12 Multiple comparisons, activations on text datasets

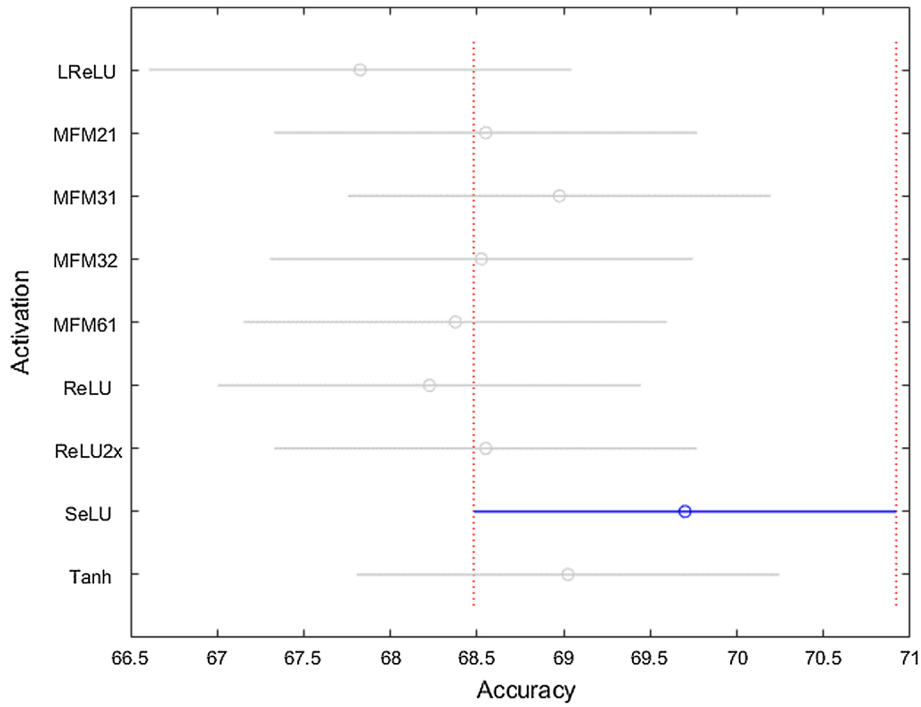

Fig. 13 Multiple comparisons, activations on medical datasets

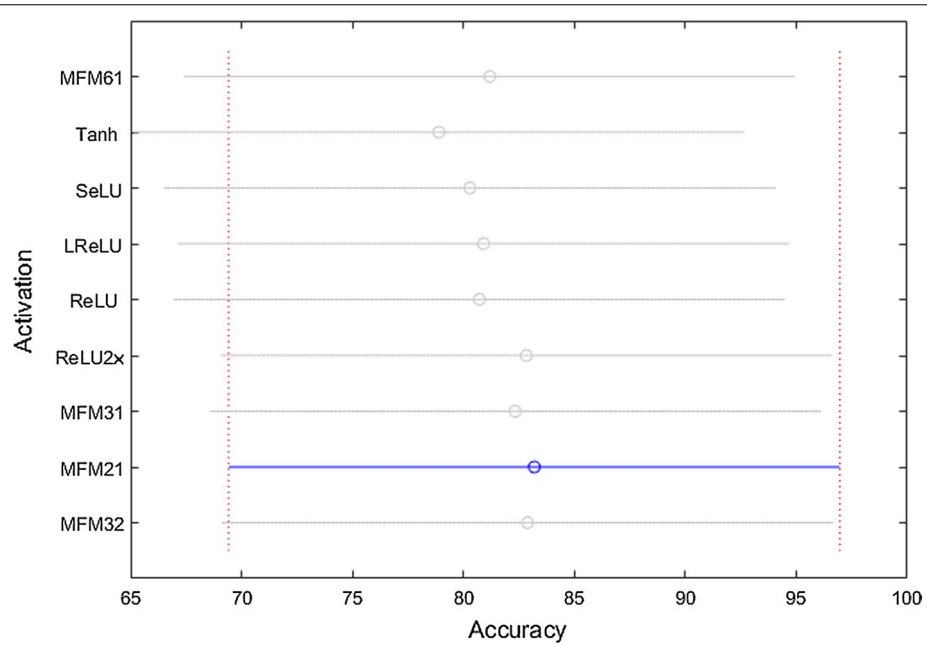

Fig. 14 Multiple comparisons, activations on sound datasets

In this study, we also performed the multiple comparison tests using Tukey's Honestly Significant Difference (HSD) test to further investigate these results. The HSD is a statistical test comparing the mean value of the performance measure for the

different activation functions. All tests of statistical significance utilize an $\alpha = 0.05$. Two activation functions with the same block letter are not significantly different with 95% statistical confidence (e.g. group a is significantly different than group b). In Table 8, the letters in the third column indicate the HSD grouping of the activation accuracy. That is, if two activations have the same letter in the HSD column, their accuracies are not significantly different.

On the image datasets, the HSD test shows ReLU6x is significantly better than the rest of the activations, and ReLU3x is better than the maxout activations. Based on these results, we see that the activations using the most memory are at the top, and maxout methods together with ReLUx are better than ReLU, LReLU, SeLU, and tanh. ReLU6x reported the highest average 100 batches training time (last column in Table 8) of 13.56 s, 1.56x higher than ReLU3x. This is expected as there are three times more layers to process. Maxout activation functions have a lower training time than ReLU3x and ReLU6x but have a higher training time than the traditional activation functions.

On the LFW dataset, the highest accuracy was reported by ReLU6x with 79.67%. It also had the highest average 100 batches time with 26.75 s. The lowest time of 2.07 s was reported by ReLU. The SeLU recorded the highest accuracy on the MS-Celeb dataset with 97.5%, but on average, for both face datasets maxout 3-2 achieved the highest average accuracy of 87.25% (Fig. 15).

On the MS-Celeb dataset, maxout 3-2 had the highest average 100 batches time of 156.94 s, and ReLU had the lowest of 7.92 s (the 100 batches time and 100 batches training time on each dataset are not displayed in a table). ReLU3x, ReLU6x and maxout 6-1 were not evaluated on the MS-Celeb dataset. The HSD test on the face datasets (Table 9) shows maxout 3-2 with the highest average accuracy, which is statistically better than LReLU, SeLU, ReLU6x, ReLU3x, and maxout 6-1. Although the LFW and MS-Celeb datasets contain face images, verification and identification tasks are different. Consequently, activation functions reported opposite results on both datasets. The extreme was SeLU which logged the highest accuracies on the MS-Celeb dataset, but the lowest on the LFW dataset.

Table 8 Activation HSD test on image datasets

Activation	Average accuracy (%)	Accuracy HSD	Average 100 batches time (s)	Average 100 batches training time
ReLU 6X	85.19	a	0.25	13.56
ReLU 3X	82.27	b	0.15	8.68
Maxout 6-1	80.68	c	0.19	9.84
Maxout 3-1	80.11	cd	0.12	6.24
Maxout 3-2	79.77	cd	0.19	8.25
ReLU 2X	79.41	cd	0.09	5.12
Maxout 2-1	78.99	d	0.11	5.24
Tanh	74.97	e	0.08	4.34
SeLU	74.76	e	0.08	2.82
ReLU	74.52	e	0.08	4.72
LReLU	74.47	e	0.08	4.52

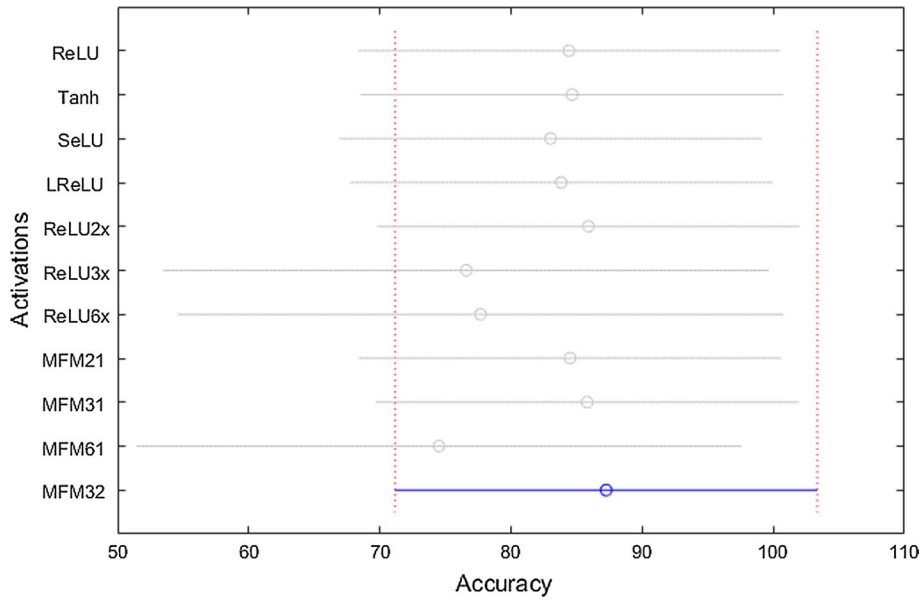

Fig. 15 Multiple comparisons, activations on face datasets

Table 9 Activation HSD test on face datasets

Activation	Average accuracy (%)	Accuracy HSD	Average 100 batches time (s)	Average 100 batches training time
Maxout 3-2	87.25	a	83.27	2425.27
ReLU 2X	85.89	ab	11.16	827.20
Maxout 3-1	85.81	ab	39.14	2659.74
Tanh	84.66	ab	5.02	700.30
Maxout 2-1	84.49	ab	31.81	3139.62
ReLU	84.41	ab	4.99	459.75
LReLU	83.82	b	9.31	342.52
SeLU	83.01	b	20.58	2126.11
ReLU 6X	77.67	c	26.75	1418.12
ReLU 3X	76.58	c	8.60	485.90
Maxout 6-1	74.50	c	11.45	521.32

In the face group, a higher training time did not translate into a higher accuracy. A total of five activation functions had a higher average training time than ReLU2x. All maxout functions except for maxout 6-1 reported the highest average 100 batch training time.

The combined results of the image and face datasets are presented in Table 10. Although ReLU6x was not tested on the MS-Celeb dataset, it is again statistically better than the rest of the activations, and its average training time is lower than any maxout activation, except for maxout 6-1. Maxout activations are statistically better than the traditional activation functions. On average, SeLU reported a low average

Table 10 Activation HSD test on image and face datasets

Activation	Average accuracy (%)	Accuracy HSD	Average 100 batches time (s)	Average 100 batches training time
ReLU 6X	84.40	a	5.55	161.41
ReLU 3X	81.68	b	1.84	58.91
Maxout 3-2	81.20	b	27.88	468.64
Maxout 3-1	81.19	b	13.13	511.67
ReLU 2X	80.65	bc	3.78	161.70
Maxout 2-1	80.04	c	10.67	602.27
Maxout 6-1	80.02	c	2.44	63.68
Tanh	76.82	d	1.73	136.90
ReLU	76.41	d	1.72	91.39
SeLU	76.33	d	6.92	407.26
LReLU	76.25	d	3.16	68.90

Table 11 Activation HSD test on text datasets

Activation	Average accuracy (%)	Accuracy HSD	Average 100 batches time (s)	Average 100 batches training time
ReLU 2x	90.41	a	15.57	594.15
Maxout 3-2	90.35	a	62.67	1485.35
ReLU	90.26	ab	7.41	349.52
Maxout 3-1	90.19	ab	29.67	972.79
Maxout 2-1	89.97	ab	17.50	440.28
Maxout 6-1	89.89	ab	48.83	1866.42
LReLU	89.71	b	13.86	754.60
Tanh	87.57	c	7.45	242.19
SeLU	83.81	d	12.20	229.62

accuracy but obtained the highest on the face identification task (Table 7). This suggests SeLU is efficient for this task.

The HSD test on text datasets (Table 11) shows six activations are statistically indistinguishable from one another (they all have the block letter 'a' in the HSD column). ReLU2x scored the highest or second highest accuracy on all text datasets except Yelp1M, while maxout 3-2 recorded within the top three accuracies except on the Sentiment140 and Yelp1M datasets. All maxout and ReLU activations are not significantly different from each other but they are statistically different from tanh and SeLU. Maxout activations took longer to train than ReLU and ReLU2x by our average epochs multiplied by the average 100 batches time metric, which considers both the computational cost and time to converge of a model. The lowest average 100 batches time was ReLU with 7.41 s, which surprisingly delivered the third highest average classification accuracy.

On the medical datasets, all the maxout and ReLU variants had a similar performance, and SeLU performed better than maxout 6-1, ReLU and LReLU (Table 12). The NN architecture for the medical datasets only had two layers, and more layers did not improve or change the activation's performance. Consequently, the performance is very similar in a shallow architecture, and this is reflected in the HSD test.

Table 12 Activation HSD test on medical datasets

Activation	Average accuracy (%)	Accuracy HSD	Average 100 batches time (s)	Average 100 batches training time
SeLU	69.70	a	0.12	13.01
Tanh	69.02	ab	0.11	22.62
Maxout 3-1	68.97	ab	0.14	24.77
Maxout 2-1	68.55	ab	0.12	21.50
ReLU 2x	68.55	ab	0.11	24.66
Maxout 3-2	68.52	ab	0.19	30.79
Maxout 6-1	68.37	b	0.13	25.05
ReLU	68.22	b	0.11	23.53
LReLU	67.82	b	0.12	27.15

Table 13 Activation HSD test on sound datasets

Activation	Average accuracy (%)	Accuracy HSD	Average 100 batches time (s)	Average 100 batches training time
Maxout 2-1	83.19	a	11.59	983.18
Maxout 3-2	82.88	ab	39.99	2839.32
ReLU2x	82.83	ab	10.68	1299.48
Maxout 3-1	82.32	ab	19.08	1619.91
Maxout 6-1	81.18	abc	30.32	3219.95
LReLU	80.90	abc	8.99	1180.16
ReLU	80.72	abc	5.05	602.98
SeLU	80.29	bc	7.87	539.84
Tanh	78.88	c	5.07	487.91

A higher training time did not translate into a higher accuracy. Maxout and ReLU activations had a higher training time than SeLU. The lowest average 100 batches training time was SeLU with 13.01 s. Although ReLU reported a low average 100 batches time, it took many epochs to converge, contrary to SeLU that converged faster than any other activation function on the NN architecture for the medical datasets. SeLU reported the fastest training time. On average, SeLU tends to converge 1.7× faster than tanh, and 2.3× faster than maxout 3-2, where maxout 3-2 reported the slowest training time across all activations.

On the sound datasets, the HSD test (Table 13) shows that LReLU, all maxout, and ReLU activations are not significantly different from each other. Maxout 2-1 performed better than tanh and SeLU. Maxout 6-1 reported the highest average 100 batches training time with 3219.95. It also, recorded the fifth highest average accuracy, while maxout 2-1 was 3.27× faster than maxout 6-1 and recorded the highest average accuracy. Maxout 3-2 converged faster than maxout 6-1. Although maxout 3-2 reported the highest average 100 batches time, the average 100 batches training time was lower than that of maxout 6-1, and maxout 3-2 recorded the second highest average accuracy. SeLU reported the second fastest average 100 batches training time, but also the second lowest average accuracy. On average, maxout 2-1 tends to converge 1.32× faster than ReLU2x.

Table 14 Activation HSD test on all datasets

Activation	Average accuracy (%)	Accuracy HSD	Average 100 batches time (s)	Average 100 batches training time
ReLU 6X	84.40	a	5.55	161.41
ReLU 3X	81.68	b	1.84	58.91
Maxout 3-2	78.74	c	33.41	813.15
Maxout 3-1	78.60	c	15.83	538.38
ReLU 2X	78.34	cd	7.83	309.25
Maxout 2-1	78.08	cd	9.90	410.16
Maxout 6-1	77.71	d	19.21	723.22
ReLU	76.42	e	3.70	166.72
Tanh	76.19	e	3.52	152.22
LReLU	76.14	e	6.43	291.52
SeLU	74.78	f	6.13	254.32

The HSD test on all datasets (Table 14) shows ReLU6x is significantly better than the rest of the activations, and ReLU3x is better than maxout and traditional activations. Although ReLU6x and ReLU3x were only evaluated on the image and LFW datasets, the HSD test suggests these two ReLU variants could provide higher accuracies than maxout or ReLU2x activation functions. Maxout activations and ReLU2x reported a similar performance, but maxout 3-1 was the only variant that did not reach the top performance in any of the experiments conducted in this study. On average, SeLU reported the lowest average accuracy, but recorded the highest accuracy with the MS-Celeb, IDMT-SMT-Audio-Effects, and three Medicaid datasets. This indicates an activation's performance may vary across data domains.

In terms of training time, maxout 3-2 reported the highest. Maxout 2-1 reported the lowest of all maxout variants, but its time was still higher than any of the ReLU variants. The average training time of ReLU6x and ReLU3x is very low as they were only tested on the image and LFW datasets. ReLU2x on average performed better than the rest of the activations on the text and audio datasets. Its average training time is above those of the traditional activation functions (Fig. 16).

We can categorize the activation functions into three groups: maxout activations as the first group, ReLU, Tanh, LReLU and SeLU (low memory usage activation functions) as the second group and ReLU2x, ReLU3x and ReLU6x (higher memory usage activation functions) as the third group. The HSD test on all datasets (Table 15) shows ReLU2x, 3X and 6X are statistically better than the rest of the activation functions. Evaluating the results across all the studied datasets, we observe that the higher the memory usage the higher the accuracy. When evaluating the training time, we observe that traditional and higher memory usage activation functions are statistically faster than the maxout activation functions. On average, the maxout activation functions reported the slowest average training time; this is expected as the number of operations is greater compared to the rest of the activation functions.

The current literature comparing maxout and other activation functions lacks to consider if an increase in the number of convolutional filters in ReLU networks, surpasses the performance of any maxout variant. Experiments that would use more than

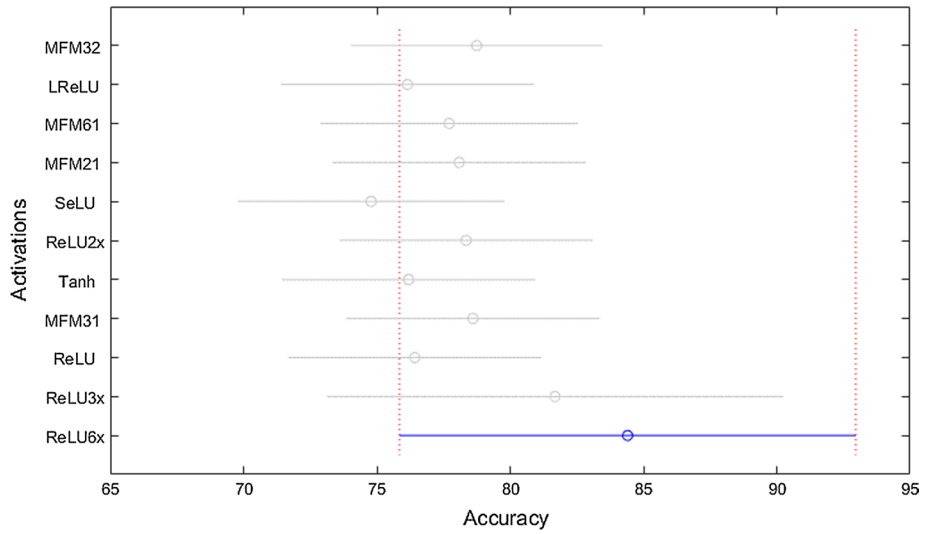

Fig. 16 Multiple comparisons, activations on all datasets

Table 15 HSD test on maxout, low memory usage activation functions and ReLU2x, 3x and 6x

Activation	Average accuracy (%)	Accuracy HSD	Average 100 batches training time	Average 100 batches training time HSD
ReLU 2X, 3X and 6X	80.22	a	229.61	a
Maxout 2-1, 3-1, 3-2 and 6-1	78.29	b	621.18	b
ReLU, Tanh, LReLU and SeLU	75.91	c	215.17	a

the maximum allowed memory per GPU were outside the scope of this work. Although ReLU6x was not tested on all datasets, our experiments provided evidence that is significantly better in terms of accuracy than any maxout variant. Furthermore, ReLU6x's average training time is lower than that of any maxout variant, even though ReLU6x contains the highest number of trainable parameters on all the tested datasets (Table 3).

Conclusion

The results from the image datasets indicate that sextupling the number of convolutional filters on ReLU performed better than the rest of the activation functions, but made training more difficult due to large number of parameters. On the sentiment classification task, ReLU2x and maxout 3-2 are likely to produce the highest classification accuracy results compared to the rest of the activations analyzed in this study. On the audio datasets, our experiments suggest that given the sound recognition task on a CNN architecture and the conversion of sounds into spectrograms, based on performance values, maxout 2-1 is likely to produce the best classification accuracy results. It is important to note that the difference between ReLU and ReLU2x is a tunable hyperparameter. The maxout activations performed better than ReLU activation functions only on the Amazon, Medicare Part D, and Google speech commands

datasets. On the medical fraud detection task, our findings indicate that given the medical fraud detection task on a NN architecture, SeLU is likely to produce the highest classification accuracy results compared to the rest of the activation functions analyzed in this study. SeLU also demonstrated its efficacy on the facial identification task, as it recorded the highest average accuracy.

Across all datasets, maxout variants provided a better average accuracy than ReLU, LReLU, SeLU and tanh, but on average the training time is slower. Results indicate that ReLU, with more filters, was the top performer, with the tradeoff of high memory usage. On average, ReLU2x converges 2.62× faster than maxout 3-2 but it is 1.85× slower than ReLU. There is no relationship between the activation functions that have a higher training time and the classification accuracy performance, but clearly adding more convolutional filters enhanced ReLU. Due to high performance and fast training relative to other top performing activations, ReLU6x is the recommended activation function for image datasets, and ReLU2x for text datasets. On the sound datasets, maxout 2-1 is the recommended activation function. Our results suggest the higher the memory usage the higher the accuracy. On average, ReLU6x will use 17.78 times more trainable CNN parameters than maxout 2-1, thus indicating a higher memory usage (Table 3) for ReLU6x.

Future work will involve conducting additional empirical studies with ReLU3x and ReLU6x on big data and hyperparameter tuning recommendations that were outside the scope of this work. Also, future work could include additional deep network architectures and domains.

Abbreviations

ANOVA: analysis of variance; CIFAR: Canadian Institute for Advanced Research; CNN: convolutional neural network; DMEPOS: durable medical equipment, prosthetics, orthotics and supplies; DNNs: deep neural networks; GSC: google speech commands; HCPCS: healthcare common procedure coding system; HSD: honestly significant difference; LEIE: list of excluded individuals and entities; LFW: labeled faces in the wild; LR: logistic regression; LRELU: leaky rectified linear unit; MFM: max-feature-map; MIN: maxout network in network; MNIST: mixed national institute of standards and technology; NIN: network in network; NN: neural networks; NPI: national provider number; PRIM: patient rule induction method; PUF: public use file; RELU: rectified linear unit; RESECH: rectified hyperbolic secant; ROC: receiver operating characteristic; RRELU: randomized leaky rectified linear unit; RUS: random under-sampling; SELU: scaled exponential linear unit; TANH: hyperbolic tangent.

Acknowledgements

The authors would like to thank the anonymous reviewers for their constructive evaluation of this paper, and also the reviewers in the Data Mining and Machine Learning Laboratory at Florida Atlantic University. Additionally, we acknowledge partial support by the NSF (CNS-1427536). Opinions, findings, conclusions, or recommendations in this paper are solely of the authors' and do not reflect the views of the NSF.

Authors' contributions

GC performed the primary literature review, analysis for this work and drafted the manuscript. TMK worked with GC to develop the article's framework and focus. PM executed the tests and helped to finalize this work. All authors read and approved the final manuscript.

Funding

Not applicable.

Availability of data and materials

The datasets used during the current study are available from the corresponding author on reasonable request.

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Received: 5 May 2019 Accepted: 23 July 2019
Published online: 03 August 2019

References

1. Delalleau O, Bengio Y. Shallow vs. deep sum-product networks. In: Advances in neural information processing systems. 2011. p. 666–74.
2. Sze V, Chen Y, Yang T, Emer J. Efficient processing of deep neural networks: a tutorial and survey. Proc IEEE. 2017;105(12):2295–329. https://doi.org/10.1109/JPROC.2017.2761740.
3. Nwankpa C, Ijomah W, Gachagan A, Marshall S. Activation functions: comparison of trends in practice and research for deep learning. 2018. arXiv:1811.03378.
4. Nair V, Hinton G. Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10). 2010.
5. Krizhevsky A, Sutskever I, Hinton G. Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. 2012. p. 1097–105.
6. Li Y, Ding P, Li B. Training neural networks by using power linear units (PoLUs). 2018. arXiv:1802.00212.
7. Ramachandran P, Zoph B, Le Q. Searching for activation functions. In: Sixth international conference on learning representations (ICLR), Vancouver. 2018.
8. Severyn A, Moschitti A. Unitn: Training deep convolutional neural network for twitter sentiment classification. In: Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015). 2015. https://doi.org/10.18653/v1/s15-2079.
9. Li J, Ng W, Yeung D, Chan P. Bi-firing deep neural networks. Int J Mach Learn Cybern. 2014;5(1):73–83.
10. Zhao H, Liu F, Li L, Luo C. A novel softplus linear unit for deep convolutional neural networks. Appl Intell. 2017;48(7):1707–20. https://doi.org/10.1007/s10489-017-1028-7.
11. Liew S, Khalil-Hani M, Bakhteri R. Bounded activation functions for enhanced training stability of deep neural networks on visual pattern recognition problems. Neurocomputing. 2016;216:718–34. https://doi.org/10.1016/j.neucom.2016.08.037.
12. Sodhi S, Chandra P. Bi-modal derivative activation function for sigmoidal feedforward networks. Neurocomputing. 2014;143:182–96. https://doi.org/10.1016/j.neucom.2014.06.007.
13. Nambiar V, Khalil-Hani M, Sahnoun R, Marsono M. Hardware implementation of evolvable block-based neural networks utilizing a cost efficient sigmoid-like activation function. Neurocomputing. 2014;140:228–41. https://doi.org/10.1016/j.neucom.2014.03.018.
14. Goodfellow I, Warde-Farley D, Mirza M, Courville A, Bengio Y. Maxout Networks. In: Proceedings of the 30th international conference on machine learning (ICML 2013). 2013.
15. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res. 2014;15(1):1929–58.
16. Wu X, He R, Sun Z, Tan T. A light CNN for deep face representation with noisy labels. IEEE Trans Inf Forensics Secur. 2015;13(11):2884–96. https://doi.org/10.1109/tifs.2018.2833032.
17. Chang J, Chen Y. Batch-normalized maxout network in network. 2015. arXiv:1511.02583.
18. Cai M, Shi Y, Liu J. Deep maxout neural networks for speech recognition. In: IEEE workshop on automatic speech recognition and understanding 2013. P. 291–6. https://doi.org/10.1109/asru.2013.6707745.
19. Park S, Kwak N. Analysis on the dropout effect in convolutional neural networks. In: Asian conference on computer vision. 2016. p. 189–204.
20. Lecun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proc IEEE. 1998;86(11):2278–324. https://doi.org/10.1109/5.726791.
21. Krizhevsky A, Hinton G. Learning multiple layers of features from tiny images. Toronto: University of Toronto; 2009.
22. Netzer Y, Wang T, Coates A, Bissacco A, Wu B, Ng A. Reading digits in natural images with unsupervised feature learning. In: NIPS workshop on deep learning and unsupervised feature learning. 2011.
23. Jebbara S, Cimiano P. Aspect-based relational sentiment analysis using a stacked neural network architecture. In: Proceedings of the twenty-second European conference on artificial intelligence. 2016.
24. Toth L. Convolutional deep maxout networks for phone recognition. In: Proceedings of the international speech communication association (INTERSPEECH). 2014.
25. Sainath T, Kingsbury B, Mohamed A, Dahl G, Saon G, Soltau H, Beran T, Aravkin A, Ramabhadran B. Improvements to deep convolutional neural networks for LVCSR. In: IEEE workshop on automatic speech recognition and understanding (ASRU). 2013. https://doi.org/10.1109/ASRU.2013.6707749.
26. Sainath T, Kingsbury B, Saon G, Soltau H, Mohamed A, Dahl G, Rmabhadran B. Deep convolutional neural networks for large-scale speech tasks. Neural Netw. 2015;64:39–48. https://doi.org/10.1016/j.neunet.2014.08.005.
27. Yoon K. Convolutional neural networks for sentence classification. In: Conference on empirical methods in natural language processing (EMNLP). 2014.
28. Poria S, Cambria E, Gelbukh A. Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis. In: Conference on empirical methods in natural language processing. 2015. https://doi.org/10.18653/v1/d15-1303.
29. Tóth L. Phone recognition with hierarchical convolutional deep maxout networks. EURASIP J Audio Speech Music Process. 2015;2015:25. https://doi.org/10.1186/s13636-015-0068-3.
30. Tóth L. Combining time-and frequency-domain convolution in convolutional neural network-based phone recognition. In: IEEE international conference on acoustics, speech and signal processing (ICASSP). 2014. https://doi.org/10.1109/icassp.2014.6853584.

31. Deng L, Abdel-Hamid O, Yu D. A deep convolutional neural network using heterogeneous pooling for trading acoustic invariance with phonetic confusion. In: IEEE international conference on acoustics, speech and signal processing (ICASSP). 2013. https://doi.org/10.1109/icassp.2013.6638952.
32. Sainath T, Mohamed A, Kingsbury B, Ramabhadran B. Deep convolutional neural networks for LVCSR. In: IEEE international conference on acoustics, speech and signal processing (ICASSP). 2013.
33. Mikolov T, Sutskever I, Chen K, Corrado G, Dean J. Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. 2013. p. 3111–9.
34. Johnson R, Zhang T. Effective use of word order for text categorization with convolutional neural networks. In: Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: human language technologies, Denver. 2015. https://doi.org/10.3115/v1/n15-1011.
35. Xu B, Wang N, Chen T, Li M. Empirical evaluation of rectified activations in convolutional network. 2015. arXiv :1505.00853.
36. Maas A, Hannun A, Ng A. Rectifier nonlinearities improve neural network acoustic models. In: International conference on machine learning (ICML). 2013.
37. He K, Zhang X, Ren S, Sun J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision. 2015. https://doi.org/10.1109/iccv.2015.123.
38. Mishkin D, Sergievskiy N, Matas J. Systematic evaluation of convolution neural network advances on the imagenet. Comput Vis Image Underst. 2017;161:11–9. https://doi.org/10.1016/j.cviu.2017.05.007.
39. Swietojanski P, Li J, Huang J. Investigation of maxout networks for speech recognition. In: IEEE international conference on acoustics, speech and signal processing (ICASSP). 2014. https://doi.org/10.1109/icassp.2014.6855088.
40. Lin M, Chen Q, Yan S. Network in network. In: Proceedings of the international conference on learning representations (ICLR). 2014.
41. Liao Z, Carneiro G. On the importance of normalisation layers in deep learning with piecewise linear activation units. In: IEEE winter conference on applications of computer vision (WACV). 2016. https://doi.org/10.1109/wacv.2016.7477624.
42. Oyedotun O, Shabayek A, Aouada D, Ottersten B. Improving the capacity of very deep networks with maxout units. In: IEEE international conference on acoustics, speech and signal processing. 2018. https://doi.org/10.1109/icassp.2018.8461436.
43. Njikam A, Zhao H. A novel activation function for multilayer feed-forward neural networks. Appl Intell. 2016;45(1):75–82. https://doi.org/10.1007/s10489-015-0744-0.
44. Goodfellow I, Mirza M, Xiao D, Courville A, Bengio Y. An empirical investigation of catastrophic forgetting in gradient-based neural networks. In: International conference on learning representations (ICLR). 2014.
45. Zhang X, Trmal J, Povey D, Khudanpur S. Improving deep neural network acoustic models using generalized maxout networks. In: IEEE international conference in acoustics, speech and signal processing (ICASSP). 2014. https://doi.org/10.1109/icassp.2014.6853589.
46. Baziotis C, Pelekis N, Doulkeridis C. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In: Proceedings of the 11th international workshop on semantic evaluation (SemEval-2017). 2017. https://doi.org/10.18653/v1/s17-2126.
47. Zhang Y, Pezeshki M, Brakel P, Zhang S, Bengio C, Courville A. Towards end-to-end speech recognition with deep convolutional neural networks. In: Sixteenth annual conference of the international speech communication association, interspeech. 2016. https://doi.org/10.21437/interspeech.2016-1446.
48. Branting L, Reeder F, Gold J, Champney T. Graph analytics for healthcare fraud risk estimation. In: Proceedings of the 2016 IEEE/ACM international conference on advances in social networks analysis and mining. 2016. https://doi.org/10.1109/asonam.2016.7752336.
49. Sadiq S, Tao Y, Yan Y, Shyu M. Mining Anomalies in Medicare Big Data Using Patient Rule Induction Method. In: IEEE third international conference on multimedia big data (BigMM). 2017. https://doi.org/10.1109/bigmm.2017.56.
50. Herland M, Khoshgoftaar TM, Bauder R. Big Data fraud detection using multiple medicare data sources. J Big Data. 2018;5(1):29. https://doi.org/10.1186/s40537-018-0138-3.
51. Klambauer G, Unterthiner T, Mayr A, Hochreiter S. Self-normalizing neural networks. In: Advances in neural information processing systems. 2017. p. 971–80.
52. Shin HC, Orton M, Collins D, Doran S, Leach M. Organ detection using deep learning. Medical image recognition, segmentation and parsing. London: Academic Press; 2016. p. 123–53. https://doi.org/10.1016/b978-0-12-802581-9.00007-x.
53. Xiao H, Rasul K, Vollgraf R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. 2017. arXiv:1708.07747.
54. Huang G, Ramesh M, Berg T, Learned-Miller E. Labeled faces in the wild: a database for studying face recognition in unconstrained environments. 2017.
55. Viola P, Jones M. Rapid object detection using a boosted cascade of simple features. Comput Vis Pattern Recognit. 2011;34:56. https://doi.org/10.1109/cvpr.2001.990517.
56. Guo Y, Zhang L, Hu Y, He X, Gao J. MS-Celeb-1M: a dataset and benchmark for large scale face recognition. In: European conference on computer vision. 2016. https://doi.org/10.1007/978-3-319-46487-9_6.
57. McAuley J, Pandey R, Leskovec J. Inferring networks of substitutable and complementary products. In: Proceedings of the international conference on knowledge discovery and data mining (KDD'15), Sydney, Australia. 2015. https://doi.org/10.1145/2783258.2783381.
58. Heredia B, Khoshgoftaar TM, Prusa JD, Crawford M. Integrating multiple data sources to enhance sentiment prediction. In: 2016 IEEE 2nd international conference on collaboration and internet computing (CIC). 2016. https://doi.org/10.1109/cic.2016.046.
59. Prusa JD, Khoshgoftaar TM. Training convolutional networks on truncated text. In: Proceedings of the IEEE international conference on tools with artificial intelligence. 2017. https://doi.org/10.1109/ictai.2017.00059.

60. Go A, Bhayani R, Huang L. Twitter sentiment classification using distant supervision. CS224N Project Rep Stanford. 2009;1(12):2009.
61. Centers for Medicare and Medicaid Services. Center for medicare and medicaid services. 2018. https://www.cms.gov/. Accessed 1 Nov 2018.
62. Centers for Medicare and Medicaid Services. Medicare provider utilization and payment data: physician and other supplier. 2018. https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Medicare-Provider-Charge-Data/Physician-and-Other-Supplier.html. Accessed 1 June 2018.
63. CMS National Provider Identifier Standard. 2018. https://www.cms.gov/Regulations-and-Guidance/Administrative-Simplification/NationalProvIdentStand/. Accessed 4 November 2018.
64. CMS. HCPCS—general information. 2018. https://www.cms.gov/Medicare/Coding/MedHCPCSGenInfo/index.html. Accessed 4 Nov 2018.
65. Centers for Medicare and Medicaid Services. Medicare provider utilization and payment data: part D prescriber. 2018. https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Medicare-Provider-Charge-Data/Part-D-Prescriber.html. Accessed 1 June 2018.
66. CMS. Medicare provider utilization and payment data: referring durable medical equipment, prosthetics, orthotics and supplies. 2018. https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Medicare-Provider-Charge-Data/DME.html. Accessed 4 Nov 2018.
67. Warden P. Speech commands: a dataset for limited-vocabulary speech recognition. 2018. arXiv:1804.03209.
68. Bosch J, Janer J, Fuhrmann F, Herrera P. A comparison of sound segregation techniques for predominant instrument recognition in musical audio signals. In: Proceedings of 13th international society for music information retrieval conference (ISMIR). 2012.
69. Stein M, Abeßer J, Dittmar C, Schuller G. Automatic detection of audio effects in guitar and bass recordings. In: Audio engineering society convention 128. Audio Engineering Society; 2010.
70. Zölzer U. DAFX: digital audio effects. New York: Wiley; 2011. https://doi.org/10.1002/9781119991298.
71. Hammer B. Popular datasets over time. 2019. https://www.kaggle.com/benhamner/popular-datasets-over-time/code. Accessed 31 May 2019.
72. Prusa JD, Khoshgoftaar TM. Designing a better data representation for deep neural networks and text classification. In: IEEE 17th international conference on information reuse and integration (IRI). 2016. https://doi.org/10.1109/iri.2016.61.
73. Zhang X, LeCun Y. Text understanding from scratch. Cornell University, Tech. Rep. 2015.
74. Chollet F. Keras. 2015. https://github.com/keras-team/keras. Accessed 1 Feb 2019.
75. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado G, Davis A, Dean J, Devin M, Ghemawat S. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. 2016.
76. Berenson ML, Levine DM, Goldstein M. Intermediate statistical methods and applications: a computer package approach. Upper Saddle River: Prentice-Hall, Inc; 1983. https://doi.org/10.2307/2288297.

Publisher's Note