© Universiti Tun Hussein Onn Malaysia Publisher's Office

**IJIE**

http://penerbit.uthm.edu.my/ojs/index.php/ijie
ISSN : 2229-838X e-ISSN : 2600-7916

The International Journal of Integrated Engineering

# Design and Evolution of Deep Convolutional Neural Networks in Image Classification – A Review

**Sachin S. Bhat[1*], Alaka Ananth[2], Venugopala P. S.[2]**

[1]Shri Madhwa Vadiraja Institute of Technology and Management,
Bantakal, Udupi - 574 115, Karnataka, INDIA

[2]NMAM Institute of Technology, Nitte - 574110, Karnataka, INDIA

*Corresponding Author

DOI: https://doi.org/10.30880/ijie.2023.15.01.019
Received 11 October 2021; Accepted 24 February 2023; Available online 28 February 2023

**Abstract:** Convolutional Neural Network(CNN) is a well-known computer vision approach successfully applied for various classification and recognition problems. It has an outstanding power to identify patterns in 1D and 2D data. Though invented in 80's, it became hugely successful after LeCun's work on digit identification. Several CNN based models have been developed to record splendid performance on ImageNet and other databases. Ability of the CNN in learning complex features at different hierarchy from the data had made it the most successful among deep learning algorithms. Innovative architectural designs and hyper parameter optimization have greatly improved the efficiency of CNN in pattern recognition. This review majorly focuses on the evolution and history of CNN models. Landmark CNN architectures are discussed with their categorization depending on various parameters. In addition, this also explores the architectural details of different layers, activation function, optimizers and other hyperparameters used by CNN. Review concludes by shedding the light on the applications and observations to be considered while designing the network.

**Keywords:** Convolutional Neural Network, Image Classification, hyper parameter, ReLU, ImageNet

## 1. Introduction

Machine Vision is a specialized area of research in Artificial Intelligence, which gives intelligence to computer systems to extract the features from data to make certain predictions without being programmed. Various Machine Learning(ML) techniques were developed in the last twenty years with an ambition to equal or excel at the human level of perception. But these systems are unsuccessful to give the human level of satisfaction [1-6]. The ambitious virtue of ML algorithms has developed a unique type of Artificial Neural Networks(ANNs) by imitating the human neuron system [7]. These algorithms are called Convolutional Neural Networks(CNNs).

The earliest CNN model was proposed by Hubel based on the cerebral cortex system [8]. But it became popular only after LeCuN's work on time series data in 1989[9]. CNNs are still considered to be the best method to empathize the image data. They have surpassed almost all the ML algorithms in pattern recognition, classification, and extraction [10]. Every major company in the IT sector has its research team to explore the new models and possible options of CNN [11]. CNN models are hot favorites in most machine vision competitions.

The architecture of CNN consists of many learning layers like convolutional, activation, pooling, etc. [12]. Convolutional layers perform many convolution operations with the help of filter banks. This helps the model in extracting and learning the local features of an image. The output of convolution layer is fed to the activation layer. This introduces non-linearity and makes the network to learn separate features from individual filters. Activation also facilitates differentiating the images based on local features. Output of activation is a sampling layer which reduces the overall dimension of the input which will be passed through other similar convolutional blocks [13]. In general, CNN

*Corresponding author: sachinbhat88@gmail.com

extracts multiple features of the image without being explicitly specified. This ability eliminates the need of separate feature extractors as in ML. Hence, deep learning(DL) models have a benefit over the ML algorithms to solve complex problems. Multiple layered representation in CNN give it the power to learn complex features at different levels of hierarchy [14]. The application of CNN in segmentation and classification has flourished after observing its performance improvement with increased depth in AlexNet[15]. Also, the processors with higher computational power like GPUs and TPUs have contributed immensely to the admiration of CNN in these years. Transfer learning approach proposed in 2015 is also another reason for the popularity of CNNs [16]. By this method, features learned for a large set of data belonging to a generic class can be transferred to classify a smaller group with a few other classes. They can also learn through a large amount of unlabeled data which makes it easy to classify thousands of different classes.

DCNN acts similar to the system of the human neuron. Information is first received in the retinotopic area. Lateral geniculate nucleus carry's out high-pass filtering and contrast normalization. V1-V4 parts of visual cortex perform detection via several stages [17]. Here, V1 and V2 act similar to convolution and sampling layers [18,19]. Temporal region of the cortex appears like the dense layers of CNN making illation about the scene. Backpropagation reduces the error rate by changing the filter weight resembles the response-based learning of humans.

CNNs started booming in AI from 2012 onwards. Many new models were proposed to solve a variety of problems. It is all because of the design of new-new blocks and the introduction of multiple hyperparameters to fine-tune. CNN started predominating the research industry after the fantastic functioning of Alexnet in the '*ImageNet Large Scale Visual Recognition Challenge*'(ILSVRC) competition. Zefnet enabled the visualization of layerwise features [20]. VGG was the first network to use a small-sized filter window to extract features from low spatial resolution. Most of the new architectures proposed these days are inspired by VGG. GoogleNet enclosed the idea of splitting the dataset first then transforming and merging. InceptionV3 gave the idea of creating multiple branches within a layer [21]. ResNet introduced jump the connection concept which later inspired many following networks. Densenet[22] and Xception[23] have explored the option of piling hundreds of layers one after another to solve complex problems. Nowadays, the objective has shifted from developing a new architecture to fine-tuning and adjusting the parameters to attain the highest accuracy for the customized data.

The remaining part of the paper is arranged in the following order: Part 2 discusses the historical aspect and evolution of CNN along with their innovative architectures. Part 3 sheds light on the layer wise details of the CNN. Some of the modern activation functions and optimizers are also briefed in this category. Part 4 briefs the applications of CNN and the challenges faced in the design. We conclude in part 5.

## 2. Evolution of CNN

Since their inception in 2012, dozens of surveys have been carried out on CNNs to compile their basic components [55]. Few papers have concentrated on their applications as well [24]. CNN has shown unparalleled efficiency in all computer vision-related applications. It is inspired by human neurons whose history begins with the experiments of Hubel in 1962. They developed many unsupervised models which collectively got the shape of CNN in the early 90's. LeCun first applied the concept of neo-cognitions in identifying the shape of digits [25]. This supervised training using backpropagation was the earliest reported work on two-dimensional CNN [9, 26]. This exhibited the quality of auto-feature extraction which was not present in the ML algorithms. Modified CNN was later developed by the name LeNet-5 which incorporated feature detection from raw images and data augmentation. It had a total of 6 layers comprising of Convolutional, Pooling, and 2 Fully-connected(FC) layers. NNs so far treated individual pixels as separate features which used to elongate the computational time. A mechanism to speed up the processor was not available. But LeNet considered the correlation between the pixel and it's neighboring to extract the features. This simultaneously boiled down the time and number of parameters. Even though this got a fame of being an inception model in CNN, its commercial ability was restricted to hand gesture recognition.

After LeNet, not much research was carried out on CNN for next 10 years. Inability to decrease the training error was the main reason incorporated. Also, it was assumed that handcrafted features from ML techniques were more efficient than the features learned through backpropagation. Hence, Support Vector Machine (SVM) gained huge popularity and overtook CNN in this period [27, 28]. The only noticeable research in this decade was by Simard who implemented CNN on MNIST database [29]. This result outperformed many ML techniques available at that time. It also showed the possibility of CNN in many other areas such as text classification, segmentation, and computer vision.

There were other problems in implementing deep learning algorithms. The architecture itself was too complex. It was taking weeks together to train the entire model on large datasets in normal computers. Fine-tuning parameters increased this time by multiple folds. In 2006, Hinton offered to train the network in a layer wise manner [30]. Huang et. al proposed the use of maxpooling to learn the invariant features [31]. GPUs were released to the market in the next year. NVIDIA introduced CUDA platform in 2007[32,33]. This regenerated the interest of researchers in CNN. ImageNet dataset was introduced by Stanford research group in 2011, which contained millions of images with labels [34]. ILSVRC annual competitions were organized to classify this data. This competition was one of the toughest in object recognition and became a major break CNN ever anticipated.

Landmark CNN architecture was contributed by AlexNet which bagged the first prize in ILSVRC2012. AlexNet surmounted the then all ML-DL algorithms. This showed the research community that the main drawback faced by CNN

so far was the absence of training data and a parallel computing process. The number of layers in AlexNet was increased to 7. The filter was an 11*11 mask. Activation functions were introduced. This helped the network to employ more diverse applications. The main problem associated with this network was overfitting. To overcome this problem, dropout layers were familiarized. Later, ZefNet[20] enabled the visualization of filter weights in individual layers. The motivation behind ZefNet was to understand the neurons in FC layers envisioning the information. Reduction in the convolution stride made more information to pass through the layers. In the later year, model of Vision Geometry Group (VGG) made the size of filters so small with increased number of layers. This development is considered one of its best in terms of research in CNN [35]. Depth is raised from 6 to 16 as well as 19 layers. Kernel mask was reduced from 11X11 and 7X7 to 3X3. This cut down the computation time and raised the parameters calculated. Pooling and zero padding were added to the network. This network shot into renown with no time due to its architectural simplicity. GoogleNet changed the overall layer design to decrease the computational cost [36]. This introduced a split and merging technique to get both local and global features. Each layer was split into different sub-blocks. Multiple kernel sizes were used instead of a single one. Hence, this was also called Network-of-networks. Average pooling with RMSprop was utilized by this [37]. Primary hindrance with GoogleNet was its highly customized behaviour to a specific module. The research from 2012 to 2016 showed that incrementing the number of layers helped CNN to discern more classes. This is mainly because of learning a variety of features at different layers and also splitting complex issues into smaller modules.

Post 2016, the research community shifted its goal from designing the layers to improve the performance by fine-tuning various parameters. One of the main concerns was dealing with the vanishing gradient problem. Cross channel learning was introduced to tackle this issue [38]. A similar concept called skip-connection was acquainted [39]. This regulated the motion of information from one layer to another. The same theme was borrowed by ResNet train till 150 layers [40]. Cross-channel connectivity was later extended to multilayer connectivity by DenseNet to enhance the representation [41]. Hybrid models were also proposed using already existing models to improve the accuracy [42,43,44]. These diverse architectures can be broadly watershed into 7 different categories as shown in the figure 1. These branches are mainly depending on the spatial exploitation, the number of layers, cross connectivity, width of individual layers, features extracted, channel boosting, and selection of applicable features.

Spatial exploitation deals with the selection of filters, stride, and activation as CNN bears a huge number of parameters to fine-tune. Depth related models tried to get more accuracy with higher depths. In conventional nets, accuracy gradually decreased after the introduction of more than ten FC layers. But HighwayNet showed a steady convergence rate till hundreds of layers [45]. ResNext tried to extend this concept with 152 layers and won ILSVRC15. The third variety namely multipaths proposed cross connection between different layers. They skipped certain layers in between while training. Dropouts were used to turn off a few nodes in each iteration. This types of networks like DenseNet mainly helped in tackling diminishing gradient issue.

As the number of layers increased by multiple folds, it was observed that some layers in them were not learning the useful features at all. Though with increased complexity, this did not contribute much towards efficiency. WideNets helped to solve this problem [46]. Though WideNets used double the parameters as Resnet, they trained more effectively than the ResNets with twice the depth. Stochastic depths assisted in reducing vanishing gradients. In the later stage, researchers trained the CNNs to learn multiple types of features rather than one type. Features played an important discrimination factor in the performance of CNN. Based on the inputs, feature map CNNs infused diverse features from the data. SqueezeNets suppressed insignificant features and provided more weight factors to more significant ones [47]. All the CNN architectures designed so far are given in table 1.

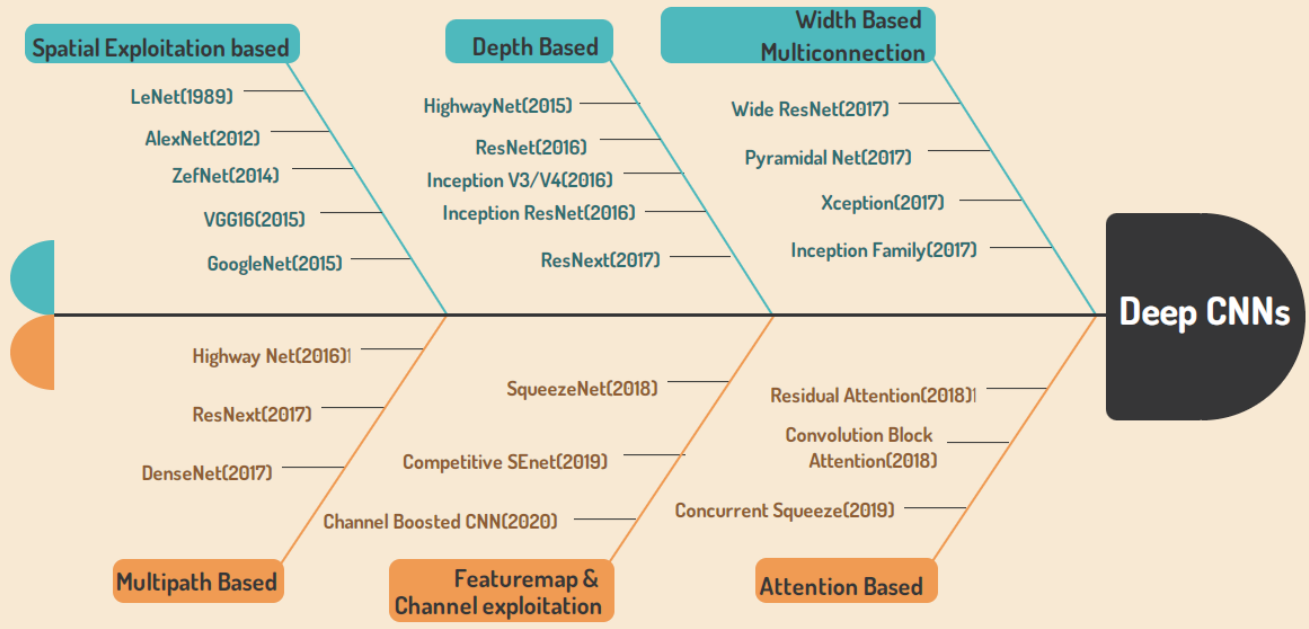


**Fig. 1 - Categories of DCNN**

**Table 1 - Various CNN architectures**

| Name | Remark | Params in lakhs | Error Rate On ImageNet | Depth | Category | Activation function |
|---|---|---|---|---|---|---|
| LeNet[9] | - 1st famous model | 0.60 | 0.80 MNIST: 0.94 | 6 | Spatial | None |
| AlexNet[15] | - Deeper and wider<br>- Used activation and Dropout | 600 | 15.3 | 7 | Spatial | ReLU |
| ZefNet[20] | - layerwise feature visualisation | 600 | 14.7 | 8 | Spatial | ReLU |
| VGG[35] | - Most famous so far<br>- Small kernel | 1380 | 7.4 | 16 -19 | Spatial | ReLU |
| GoogLeNet [36] | - Split and Merge<br>- blocks introduced | 40 | 6.70 | 21 | Spatial | ReLU |
| InceptionV3 [48] | - Replacement of larger filter by smaller one introduced 1D filter | 236 | 3.58 | NA | Depth | ReLU |
| Highway Networks [45] | - Multiple Paths | 23 | 7.8 | 19 | Depth + Cross connection | ReLU and tanh |
| InceptionV4 [21] | - Split and Merge<br>-Uses asymmetric filter | - | 3.80 | NA | Depth | ReLU |
| Inception [36] | - Split, Transform, Merge, and Residual Links | - | 3.11 | NA | Depth + Cross connection | ReLU |
| ResNet [40] | - Residual Learning and Identity mapping based skip connection | 680 | 3.57 | 152 | Spatial + Depth + Cross connection | ReLU |
| DelugNet [49] | - Allow cross layer data flow | 2020 | 3.76 | 146 | Cross connection | ReLU |
| FractalNet [50] | - Multiple path lengths | 3860 | 7.27 | 20 | Cross connection | ReLU |
| WideResNet [46] | - More width, less depth | 3650 | 3.89 | 28 - | Width | ReLU |

| | | | | | | |
|---|---|---|---|---|---|---|
| Xception [51] | - Depth wise CN followed by point wise CN | 2280 | 0.06 | 36 | Width | ReLU, ELU |
| RANN [52] | - Introduces Attention Mechanism | 860 | 3.90 | 452 | Attention | Sigmoid |
| ResNexT [22] | - Cardinality<br>- Homogeneous topology | 6810 | 3.58 | 29 | Spatial | ReLU |
| SE Net [47] | - Convolution in group<br>-Dependency between feature maps | - | 3.60 | 154 | Feature Map | ReLU |
| DenseNet [41] | - Cross-layer information flow | 153 | 5.19 | 250 | Cross connection | ReLU |
| PolyNet [40] | - diversity in structure<br>- Poly Inception Module | - | 4.25 | - | Width | ReLU |
| PyramidNet [23] | Gradual increase of width | 270 | 4.7 | 165 | Width | ReLU |
| Convolution Block Attention Module [23] | - combination of spatial and feature map information | 4896 | 5.60 | 100 | Attention | ReLU |
| Squeezenet [47] | -Channelwise excitation | - | 12 | - | Attention | ReLU |
| ChannelBoosted CNN [53] | - Boosting with extra information | - | - | - | Channel Boost | - |
| Competitive SEnet [54] | Channel rescaling | 3692 | 3.58 | 28 | Feature Map | ReLU, Sigmoid |

## 3. The Architecture of CNN Layers

In image analysis and classification, DCNN has shown unparalleled efficiency in detecting patterns. Convolutional layers in the network differentiate CNN from other types of Neural Network algorithms. These layers receive the inputs, transform them by performing convolution, and forwards these transformed outputs to subsequent layers.

DCNN is a most popular ANN in image analysis and classification having a specialization in detecting the patterns and making sense of them. Convolutional layers hidden inside the network differentiate it from Multi-Layer Perceptron and Recurrent Neural Network. Convolution operation in these layers receives the inputs, transforms them using certain operation, and outputs the transformed input to the following layer. Apart from these layers, the network also contains pooling and a sequence of FC layers. Different types of filters are used to record the features of the model. Kernel values are updated according to the response of the model to find the best kernel value. Max pooling and activation functions are utilized for this purpose.

CNN takes an order of three tensors as its input irrespective of the type of image. These input images will be processed through a series of layers. Here, the layer denotes a single processing step. This can be a convolution layer, FC layer, activation, or pooling layer. In a forward pass, CNN goes through layers as shown in Equation 1.

$$x_1 \rightarrow p_1 \rightarrow x_2 \rightarrow \cdots \rightarrow x_{L-1} \rightarrow p_{L-1} \rightarrow x_L \rightarrow p_L \rightarrow z \quad (1)$$

The input x1 is processed in several layers and the outputs of these layers are termed as x2, x3 ... xL. Each layer's output will be used as the input for the next. Parameters involved in the processing of these layers are denoted by tensors p1, p2, .... pL. However, one extra layer ‘z’ is added for backpropagation. For the classification problem, the approach is to output xL as a C dimensional vector, where ith entry encodes the prediction. If all the parameters in the CNN model are learned, the network is run forward for prediction. In addition to this, various hyperparameters are also used to enhance performance. Following section briefs these components used in CNN.

### 3.1 Convolutional Layer

This layer acts as a feature extractor. Each neuron in a feature or activation map contains a sensory field that is connected to neurons in the previous layer through trainable weights called filter banks. The element performing the convolution operation is referred to as kernel/filter. These filters are applied on each image and images are modified according to the filter values. Subsequent feature map values hi,j are calculated according to equation 2 for input x, convolution kernel k, kernel width m, convolution output h.

$$h_{i,j} = \sum_{k=1}^{m} \sum_{l=1}^{m} w_{k,l}\, x_{i+k,j+l-1} \quad (2)$$

With the size of the filter being $m * n$, dimension of the image being $l$ and number of filters being $k$, parameters generated in each layer($p$) with bias one are calculated as

$$p = [(m * n * l) + 1] * k \quad (3)$$

The pooling layer is applied to the feature maps generated by each convolution layer to compress the spatial size of an image so that the final output of the network will be 1X1 feature vector.

## 3.2 Pooling Layer

Pooling layers were created to decrease the number of parameters expected to portray layers deeper in the network. Pooling additionally decreases the number of calculations necessary for training the network, or simply running it forward during a classification operation. Pooling layers give some restricted measures of translational and rotational invariance. Three types of pooling can be generally used: average pooling, max pooling, and min pooling [46]. Pooling passes the dominant information present in the neighborhood to the next layer depending on the type of pooling. This can be mathematically expressed as in equation 4 where *fp* is a type of pooling.

$$h_{i,j} = f_p\{x_{i+k-1,j+l-1,} \forall 1 \leq k \leq m \; and \; 1 \leq l \leq m\} \quad (4)$$

Some CNN architectures like GoogleNet have tried bigger strides in substitution of pooling.

## 3.3 Fully Connected (FC) Layer

FC layers are utilized in the last phases of the CNN to interface the yield of the convolution layer to the next level. Usually, the output of the convolution layer will be in 3D whereas FC layer anticipates a 1D vector of numbers. Hence, a flattened layer is used at the beginning of FC block to convert the output of the last pooling layer to a vector. FC layer contains some dense layers with thousands of neurons. These neurons are trained according to the input weights. The dense layers learn how to utilize the features created by convolutions to accurately classify the images. A dropout layer is used between two dense layers for regularization and to forbid overfitting. The neurons of the last layer of FC block are a linear classifier that the outputs of the network can be interpreted as posterior probabilities. This layer assigns a probability value to each class for the input to perform classification.

## 3.4 Dropout

Dropout is a regularization method for precluding overfitting of a network [57]. It is done by expelling some nodes in a layer with a defined probability *p* who do not take part in the training. Subsequent to training, they are supplanted in the network with their original weights. This forestalls the FC layers from overfitting and betters the performance on a validation set. The output of a convolution layer is a linear weighted sum of the inputs.

While looking at the ordinary LSM loss for this layer of a regular (5) and dropout network (6),

$$E_N = \frac{1}{2} (t - \sum_{i=1}^{n} p_i w_i I_i)^2 \quad (5)$$

$$E_D = \frac{1}{2} (t - \sum_{i=1}^{n} \delta_i w_i I_i)^2 \quad (6)$$

The dropout rate is $\delta$ with probability p, where

$$\delta \sim \text{Bernoulli}(p) \quad (7)$$

The backpropagation for network training utilizes a gradient descent method. By taking the derivative of the gradient of the dropout network in equation (5) we get,

$$\frac{\partial E_N}{\partial w_i} = -t p_i I_i + w_i p_i^2 I_i^2 + \sum_{j=1, j\neq 1}^{n} w_i p_i p_j I_i I_j \quad (8)$$

Now, we find the expectation of gradient of the dropout network,

$$E\left[\frac{\partial E_D}{\partial w_i}\right] = -tp_iI_i + w_ip_i^2I_i^2 + w_iVar(\delta_i)I_i^2 + \sum_{j=1,j\neq 1}^{n} w_ip_ip_jI_iI_j$$
$$= \frac{\partial E_N}{\partial w_i} + w_iVar(\delta_i)I_i^2$$
$$= \frac{\partial E_N}{\partial w_i} + w_ip_i(1\text{-}p_i)\,I_i^2 \quad (9)$$

By looking at equation 9, the gradient with dropout has the same expectation as the gradient of a regularised network. Therefore, minimising the dropout loss (in equation 6) is the same as minimising a regularised network shown in equation 10, and differentiating the regularized network will get the gradient of a dropout network.

$$E_R = \frac{1}{2}\,(t - \sum_{i=1}^{n} p_iw_iI_i)^2 + \sum_{i=1}^{n} w_ip_i(1-p_i)I_i^2 \quad (10)$$

To obtain the maximum regularization, p (1- p) in equation 10 should be maximum. This is possible only if p=0.5. Hence, in general, a dropout ratio of 0.5 will be used in most of the networks.

## 3.5 Activation Function

An important task in designing any deep learning architecture is to fine tune its hyperparameters so that the network attains maximum accuracy by incorporating the highest number of features from the data. Some of the important hyperparameters to be tuned in the network are: activation function, padding, stride size of the input, optimizer, and learning rate. A description of a few of these parameters with the incorporated methods of fine-tuning is given below.

Activation functions(AFs) decide the output of CNN like yes or no. It converts the learned linear mappings into a nonlinear type for propagation. Most of the networks suffer from either vanishing or exploding gradient problems because of the repeated multiplication of derivative terms with the values. Hence, AFs restrict the values of these gradients in a specific range. Despite the depth of CNNs, one of the important features in the architecture is the use of AFs. Many new functions have been proposed ever since the discovery of Tanh [58].

Image class and image data have a non-linear relationship. The activation function must be nonlinear for a neural network to construct this relationship. In absence of non-linearity, a neural network would be capable of only linear classification. Rectified Units family is the most notable one in this. ReLu is a piecewise linear function which clips the –ve part to zero while retaining the +ve part [63]. In contrast to this, a variation of ReLu called leakyReLu assigns a non-zero slop to –ve part [64]. In Parameterized ReLu or PReLU, coefficients of the –ve parts are adaptively learned rather than predefined [66]. In the third variant randomised ReLU or RReLU, slopes of –ve part are randomised within a range in the training phase and are fixed in testing [67]. Exponential Linear Units or ELUs have -ve values, which allows the network to push the mean activations more like zero [65]. Properties of some of the linear AFs with variants of ReLU are given in table 2.

**Table 2 - Description of different activation functions**

| Function | Description | Function representation | Error Rate |
|---|---|---|---|
| TanH[58] | Hyperbolic tangent<br>*suffers from Vanishing gradient problem*<br>Limitation of tanH birthed ReLu | $f(x) = \frac{1}{1+\exp(-1)}$<br>with range (-1,1) | NA |
| Hard Tanh[62] | Hard hyperbolic function<br>Computationally more efficient | $f(x) = \max(-1, \min(1, x))$ | CIFAR10: 0.1064 |
| Sigmoid [59] | More generalized logistic activation function used for multiclass problem<br>suffers from backpropagation error | $f(x) = \frac{1}{1+e^x}$<br>for x being input | NA |
| Hardsigmoid[60] | Used when approximation output is sufficient<br>Variation of sigmoid with lesser computation cost<br>Preferred when speed is more important than precision | $f(x) = \max(0, \min(1, x))$ | NA |
| ReLU[63] | *Most popular so far*<br>*No backpropagation errors* | $f(x) = \max(x, 0)$ | CIFAR10:0.1245<br>CIFAR00: 0.429 |

| | | | |
|---|---|---|---|
| | Suffers from vanishing gradient problem | with *range (0, ∞)* | |
| LeakyReLU[64] | ReLu with increased range<br>Both function and its derivatives are monotonic in nature | $f(x) = \max(x, ax)$<br>a=0.01 with range (-∞ to +∞) | For a=100<br>CIFAR10:0.1266<br>CIFAR00: 0.4205 |
| PReLU[66] | Parametric ReLU<br>multiplies the negative input by a small value and keeps the positive input as is | *f(x)=x if x>0, else ax*<br>*acts like ReLU for a=0*<br>*and as LeakyReLu for a>0* | CIFAR10:0.1179<br>CIFAR00: 0.4163 |
| RReLU[67] | Randomized version of leaky ReLU | In training,<br>*f(x)=x if x>0, else ax*<br>In testing<br>*f(x)=x/a* | CIFAR10:0.1119<br>CIFAR00: 0.4025 |
| ELU[65] | Can produce negative outputs unlike Relu<br>Range(-infinity to infinity) | *f(x)=x*, if x>0, else a($e^x$ – 1) | CIFAR10:0.1804<br>CIFAR00: 0.4580 |

## 3.6 Optimizer

During the training, we change the weights of the network to attempt to limit the loss function (E), and improve the accuracy of the predictions(X). Optimizers are used to discover the optimized value for the model weights (W). These are one of the two categories. First order optimizers like Gradient descent use Gradient values that minimize or maximize the loss function with respect to the parameters. Second order optimizers like Hessian use second order derivatives to minimize the loss function. Due to computational cost, second order optimizers are not widely used. Gradient Descent is the most significant method and the foundation to train and optimize DL systems. It updates the weights by controlling the variance and tunes the parameters in the direction of decreasing the loss function. It is a method of minimizing the objective function J(θ) by updating the parameters in the opposite direction of the gradient of the objective function ∇ θ. J(θ). Properties of different gradient based optimizers are given in table 3.

**Table 3 - Optimizers**

| Optimizer | Property | Accuracy on MNIST | Loss |
|---|---|---|---|
| Stochastic *Gradient Descent* (SGD)[68] | Function fluctuates heavily<br>Computes gradient of cost function<br>Performs parameter(**θ**) updation for every training example $x^i$ and label $y^i$ with a learning rate η<br>$\theta=\theta-\eta.\nabla_\theta.J(\theta, x^{(i)},y^{(i)})$<br>Deals well with sparse data. | 0.8903 | 1.886 |
| Adagrad [69] | Learning rate is always decaying<br>g(t,i) is the *gradient of the loss function with respect* to the parameter θ(i) at time step t<br>$\theta(t+1) = \theta(t) - \frac{\eta}{\sqrt{G_{t,i}}} g_{t,i}$<br>$G_{t,i}$ is a diagonal matrix which is the sum of the squares of the past gradients. | 0.4086 | 2.048 |
| RMSProps [70] | Special version of Adagrad<br>Accumulates gradients in a fixed window.<br>The only difference RMSProp has with Adagrad is that the $g_t$ term is calculated by exponentially decaying average $E[g^2]_t$ and not the sum of gradients.<br>- $\theta(t+1) = \theta(t) - \frac{\eta}{\sqrt{E[g^2]_t}} g_{t,i}$ | 0.458 | 1.649 |
| Adadelta [71] | Robust extension of Adagrad where denominator is RMS error criterion of the gradient<br>$\theta(t+1) = \theta(t) - \frac{\eta}{\sqrt{RMS[g_t]}} g_{t,i}$ | 0.272 | 2.20 |
| Adam [72] | Adagrad+RMSProps<br>Requires less memory and less tuning | 0.72 | 0.919 |

| | | | |
|---|---|---|---|
| | Rectifies every problem that is *faced in other optimization techniques*<br>$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \varepsilon}\hat{m}_t$<br>Where $v_t$ and $m_t$ are estimates of mean and uncentered variance | | |
| Adamax [72] | Variant of Adam based on the infinity norm constrained $v_t$ denoted by $u_t$<br>$\theta_{t+1} = \theta_t - \frac{\eta}{u_t}\hat{m}_t$ | 0.6263 | 1.188 |
| Nadam [73] | RMSprops with Nesterov momentum $\hat{m}_t$ and decay rate $\beta_1$<br>$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t}+\varepsilon}(\beta_1 \hat{m}_t \frac{(1-\beta_1)g_t}{(1-\beta_t^1)})$ | 0.7196 | 0.942 |

## 3.7 Padding

Padding is a group of zeros placed around the image. This is done to prevent the reduction of the image dimension after performing convolution. With repeated convolution, size of the output image will gradually shrink which can cause a problem of losing the information at image boundaries. This can be tackled by adding zeros in the input before convolution as shown in figure 2. This helps in maintaining the input dimension at the output. The padding width should meet equation 11, where *p* is padding and *f* is the filter dimension.

$$p = \frac{f-1}{2} \tag{11}$$

## 3.8 Stride

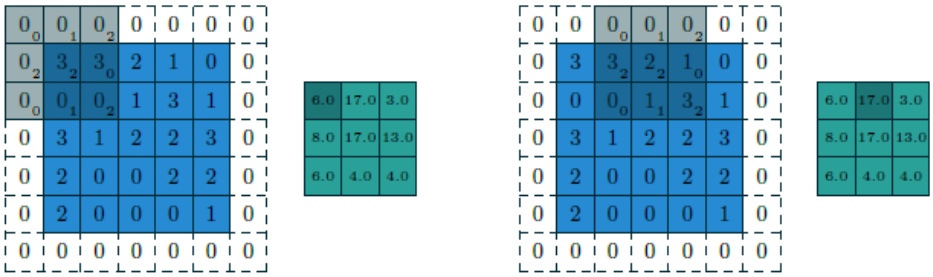


**Fig. 2 - Convolution with zero padding and stride = 2**

Stride(S) is a parameter portrays what number of pixels a filter will be interpreted horizontally. For example, GoogleNet uses stride value 2. For S=2, filter horizontally moves across the image two pixels at a time as shown in Figure 6. The dimensions of the output matrix($n_p$) - taking into account padding(p) and stride(s) can be calculated as

$$n_p = \frac{n_{in} + 2p - f}{s} + 1 \tag{12}$$

## 4. Applications and Observations

CNN is fortunately implemented in diverse DL applications like pattern recognition, classification, segmentation and, many more. But, like any other DL technique CNN too requires a large dataset. It has shown enormous achievement on labelled data. It is hugely popular in natural language processing. Many CNN-based applications are popular in Language modelling and information retrieval. Researches in [74,75] have used CNN to find word relations and matching of two lines. CNN is also having a notable contribution in face and action detection [76]. Zang et. al proposed cascaded CNN for pose estimation [77]. Sijin had shown that perceptrons in hidden layers can learn local features of the body [78]. 3D CNNs are also developed to extract features from different channels of input frames [79]. Newly region-based CNNs are extensively used for object identification. Ren et. al developed full RCNN to detect the boundary of an object at different places [80]. Region based CNNs are employed for object detection in PASCALVOC dataset. CNNs are also used in medical field for the diagnosis of lung cancer or breast cancer. Externally labelled data is used to train the network through which CNN will be able to learn the features and predict accordingly [81,82]. Apart from these, a few other major areas where CNNs have shown state of art performance are traffic detection [83], Twitter sentiment analysis [84], speech recognition [85], and vocabulary recognition [86].

CNN has attained great functioning on different types of data. However, particularly in image classification, the results are not promising with respect to the identification of pose and orientation. Data augmentation proposed by

Alexnet has solved this issue. Data augmentation assists with improving the performance of CNN in two ways. On one side it increases the number of images in the training dataset. On the other hand, it learns more features by means of rotation, scaling, replicating, and mirroring the labelled images. Features learnt by the lower layers will be passed on to the higher layers in capsule networks. Another challenge CNN faces are the noisy data. Even a small amount of noisy data is enough to confuse the network to classify the same type of images with and without noise as two separate classes. Some of the main observations to be considered while designing a network are listed below:

- As the features are automatically extracted by the CNN layers without human intervention, it is sometimes important to know their nature. Feature map visualization helps to a large extent in this direction.
- Unlike humans, CNN needs a huge amount of data to learn things. So, it is always essential to have enough labelled training data while handling DL problems.
- Selection of hyperparameter is the most important thing to take care of when designing the architecture. An inappropriate selection of hyperparameter will degrade the performance of CNN in a great magnitude. Also, a small change in the value of above said parameter can cost dearly. So, it is the main design issue to be addressed with the utmost care during the network design.
- CNN has a huge potential in handling semantic data. Meta-algorithmic techniques such as ensembles are successfully sued to a wide range of situations.
- With the advancement in GPUs and parallel computing, the learning limit of CNN is upgraded. However, the depth of the network causes a substantial overhead on memory and other resources. The primary worry with CNNs is the run-time relevance. Also, utilization of CNN is blockaded in small types of equipment like mobile in light of its high computational expense.
- Hyperparameter tuning is a repetitive and instinct-driven task, which cannot be characterized by means of explicit formulation.

## 5. Conclusion

CNN has gained exceptional ground in pattern recognition and has regenerated interest in neural networks. It has a great ability to extract the features in one and two-dimensional data. A lot of research has been done to improve the performance of CNN architectures in image classification tasks. Innovative architectural designs and hyper parameter optimizations have greatly improved the efficiency of CNN in pattern recognition. This study examines improvement in CNN architectures based on processing unit design patterns and their details. This overview covers the evolutionary history of CNNs, its applications, problems, and architectural aspects, in addition to categorizing CNNs into several classes.

This is a new step, maybe a very huge and crucial step, in the ongoing effort to improve the capabilities of computer machines so that they can solve problems that are vital to us humans. Clearly, the new CNN models have a lot of potential for making our lives easier and reducing our reliance on unreliable and expensive human resources to complete routine jobs efficiently.

## Acknowledgement

This work is supported by VTU under VTU Research Grants Scheme 2021 and Karnataka Science and Technology Academy under Short Term Studies 2022.

## References

[1] Chapelle, Olivier. "Support vector machines for image classification", 1998.
[2] Lowe, D. G. (1999, September). Object recognition from local scale-invariant features. In Proceedings of the seventh IEEE international conference on computer vision (Vol. 2, pp. 1150-1157). IEEE.
[3] Bay, H., Ess, A., Tuytelaars, T., & Van Gool, L. (2008). Speeded-up robust features (SURF). Computer vision and image understanding, 110(3), 346-359.
[4] Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection. In 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05) (Vol. 1, pp. 886-893). Ieee.
[5] Ojala, Timo, Matti Pietikäinen, and David Harwood. "A comparative study of texture measures with classification based on featured distributions." Pattern recognition Vol. 29, no. 1, pp. 51-59, 1996.
[6] Ojala, T., Pietikäinen, M., & Harwood, D. (1996). A comparative study of texture measures with classification based on featured distributions. Pattern recognition, 29(1), 51-59.
[7] LeCun, Y., Kavukcuoglu, K., & Farabet, C. (2010, May). Convolutional networks and applications in vision. In Proceedings of 2010 IEEE international symposium on circuits and systems (pp. 253-256). IEEE.
[8] Hubel, D. H., & Wiesel, T. N. (1968). Receptive fields and functional architecture of monkey striate cortex. The Journal of physiology, 195(1), 215-243.
[9] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4), 541-551.

[10] Ciresan, D., Giusti, A., Gambardella, L., & Schmidhuber, J. (2012). Deep neural networks segment neuronal membranes in electron microscopy images. Advances in neural information processing systems, 25.
[11] Deng, L., & Yu, D. (2014). Deep learning: methods and applications. Foundations and trends® in signal processing, 7(3–4), 197-387.
[12] Jarrett, K., Kavukcuoglu, K., Ranzato, M. A., & LeCun, Y. (2009, September). What is the best multi-stage architecture for object recognition? In 2009 IEEE 12th international conference on computer vision (pp. 2146-2153). IEEE.
[13] Scherer, D., Müller, A., & Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. In Artificial Neural Networks–ICANN 2010: 20th International Conference, Thessaloniki, Greece, September 15-18, 2010, Proceedings, Part III 20 (pp. 92-101). Springer Berlin Heidelberg.
[14] Bengio, Y. (2009). Learning deep architectures for AI. Foundations and trends® in Machine Learning, 2(1), 1-127.
[15] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 84-90.
[16] Qureshi, A. S., Khan, A., Zameer, A., & Usman, A. (2017). Wind power prediction using deep neural network based meta regression and transfer learning. Applied Soft Computing, 58, 742-755.
[17] Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 22(10), 1345-1359.
[18] Laskar, M. N. U., Giraldo, L. G. S., & Schwartz, O. (2018). Correspondence of deep neural networks and the brain for visual textures. arXiv preprint arXiv:1806.02888.
[19] Grill-Spector, K., Weiner, K. S., Gomez, J., Stigliani, A., & Natu, V. S. (2018). The functional neuroanatomy of face perception: from brain measurements to deep neural networks. Interface Focus, 8(4), 20180013.
[20] Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13 (pp. 818-833). Springer International Publishing.
[21] Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2017, February). Inception-v4, inception-resnet and the impact of residual connections on learning. In Proceedings of the AAAI conference on artificial intelligence (Vol. 31, No. 1).
[22] Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1492-1500).
[23] Han, D., Kim, J., & Kim, J. (2017). Deep pyramidal residual networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 5927-5935).
[24] Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., & Muharemagic, E. (2015). Deep learning applications and challenges in big data analytics. Journal of big data, 2(1), 1-21.
[25] Fukushima, K., & Miyake, S. (1982). Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In Competition and Cooperation in Neural Nets: Proceedings of the US-Japan Joint Seminar held at Kyoto, Japan February 15–19, 1982 (pp. 267-285). Springer Berlin Heidelberg.
[26] Linnainmaa, S. (1970). The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors (Doctoral dissertation, Master's Thesis (in Finnish), Univ. Helsinki).
[27] Joachims, T. (2005, June). Text categorization with support vector machines: Learning with many relevant features. In Machine Learning: ECML-98: 10th European Conference on Machine Learning Chemnitz, Germany, April 21–23, 1998 Proceedings (pp. 137-142). Berlin, Heidelberg: Springer Berlin Heidelberg.
[28] Decoste, Dennis, and Bernhard Schölkopf. "Training invariant support vector machines." Machine learning, Vol. 46, no. 1, pp. 161-190, 2002.
[29] Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web]. IEEE signal processing magazine, 29(6), 141-142.
[30] Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. Neural computation, 18(7), 1527-1554.
[31] Ranzato, M. A., Huang, F. J., Boureau, Y. L., & LeCun, Y. (2007, June). Unsupervised learning of invariant feature hierarchies with applications to object recognition. In 2007 IEEE conference on computer vision and pattern recognition (pp. 1-8). IEEE.
[32] Lindholm, E., Nickolls, J., Oberman, S., & Montrym, J. (2008). NVIDIA Tesla: A unified graphics and computing architecture. IEEE micro, 28(2), 39-55.
[33] Nickolls, J., Buck, I., Garland, M., & Skadron, K. (2008). Scalable parallel programming with cuda: Is cuda the parallel programming model that application developers have been waiting for? Queue, 6(2), 40-53.
[34] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
[35] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).
[36] Dauphin, Y., De Vries, H., & Bengio, Y. (2015). Equilibrated adaptive learning rates for non-convex optimization. Advances in neural information processing systems, 28.

[37] Karpathy, A., Johnson, J., & Fei-Fei, L. (2015). Visualizing and understanding recurrent networks. arXiv preprint arXiv:1506.02078.
[38] Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.
[39] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
[40] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).
[41] Yamada, Y., Iwamura, M., & Kise, K. (2016). Deep pyramidal residual networks with separated stochastic depth. arXiv preprint arXiv:1612.01230.
[42] Huang, G., Sun, Y., Liu, Z., Sedra, D., & Weinberger, K. Q. (2016). Deep networks with stochastic depth. In Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14 (pp. 646-661). Springer International Publishing.
[43] Targ, S., Almeida, D., & Lyman, K. (2016). Resnet in resnet: Generalizing residual architectures. arXiv preprint arXiv:1603. 08029..
[44] Srivastava, R. K., Greff, K., & Schmidhuber, J. (2015). Highway networks. arXiv preprint arXiv:1505.00387.
[45] Zagoruyko, S., & Komodakis, N. (2016). Wide residual networks. arXiv preprint arXiv:1605.07146.
[46] Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-excitation networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7132-7141).
[47] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).
[48] Kuen, J., Kong, X., Wang, G., & Tan, Y. P. (2017). DelugeNets: deep networks with efficient and flexible cross-layer information inflows. In Proceedings of the IEEE International Conference on Computer Vision Workshops (pp. 958-966).
[49] Larsson, G., Maire, M., & Shakhnarovich, G. (2016). Fractalnet: Ultra-deep neural networks without residuals. arXiv preprint arXiv:1605.07648.
[50] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1251-1258).
[51] Wang, F., Jiang, M., Qian, C., Yang, S., Li, C., Zhang, H., ... & Tang, X. (2017). Residual attention network for image classification. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3156-3164).
[52] Khan, A., Sohail, A., & Ali, A. (2018). A new channel boosted convolutional neural network using transfer learning. arXiv preprint arXiv:1804.08528.
[53] Hu, Y., Wen, G., Luo, M., Dai, D., Ma, J., & Yu, Z. (2018). Competitive inner-imaging squeeze and excitation for residual network. arXiv preprint arXiv:1807. 08920..
[54] Khan, Asifullah, Anabia Sohail, Umme Zahoora, and Aqsa Saeed Qureshi. "A survey of the recent architectures of deep convolutional neural networks." arXiv preprint arXiv:1901.06032, 2019.
[55] Lee, C. Y., Gallagher, P. W., & Tu, Z. (2016, May). Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In Artificial intelligence and statistics (pp. 464-472). PMLR.
[56] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.
[57] LeCun, Y., Bottou, L., Orr, G. B., & Müller, K. R. (1998). Neural networks: Tricks of the trade. Springer Lecture Notes in Computer Sciences, 1524(5-50), 6.
[58] Han, J., & Moraga, C. (1995). The influence of the sigmoid function parameters on the speed of backpropagation learning. In from Natural to Artificial Neural Computation: International Workshop on Artificial Neural Networks Malaga-Torremolinos, Spain, June 7–9, 1995 Proceedings 3 (pp. 195-201). Springer Berlin Heidelberg.
[59] Courbariaux, M., Bengio, Y., & David, J. P. (2015). Binaryconnect: Training deep neural networks with binary weights during propagations. Advances in neural information processing systems, 28.
[60] Elfwing, S., Uchibe, E., & Doya, K. (2018). Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. Neural Networks, 107, 3-11.
[61] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. Journal of machine learning research, 12(ARTICLE), 2493-2537.
[62] Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th international conference on machine learning (ICML-10) (pp. 807-814).
[63] Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013, June). Rectifier nonlinearities improve neural network acoustic models. In Proc. icml (Vol. 30, No. 1, p. 3).
[64] Clevert, D. A., Unterthiner, T., & Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). arXiv preprint arXiv:1511.07289.

[65] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 84-90.
[66] Xu, B., Wang, N., Chen, T., & Li, M. (2015). Empirical evaluation of rectified activations in convolutional network. arXiv preprint arXiv:1505.00853.
[67] Schaul, Tom, Ioannis Antonoglou, and David Silver. "Unit tests for stochastic optimization." arXiv preprint arXiv:1312.6055, 2013.
[68] Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. Journal of machine learning research, 12(7).
[69] Tieleman, T., & Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural networks for machine learning, 4(2), 26-31.
[70] Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701.
[71] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
[72] Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013, May). On the importance of initialization and momentum in deep learning. In International conference on machine learning (pp. 1139-1147). PMLR.
[73] Collobert, R., & Weston, J. (2008, July). A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th international conference on Machine learning (pp. 160-167).
[74] Xue, X., & Yin, X. (2011, October). Topic modeling for named entity queries. In Proceedings of the 20th ACM international conference on Information and knowledge management (pp. 2009-2012).
[75] Farfade, S. S., Saberian, M. J., & Li, L. J. (2015, June). Multi-view face detection using deep convolutional neural networks. In Proceedings of the 5th ACM on International Conference on Multimedia Retrieval (pp. 643-650).
[76] Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. IEEE signal processing letters, 23(10), 1499-1503.
[77] Lee, S., & Nirjon, S. (2020, June). Fast and scalable in-memory deep multitask learning via neural weight virtualization. In Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services (pp. 175-190).
[78] Ji, S., Xu, W., Yang, M., & Yu, K. (2012). 3D convolutional neural networks for human action recognition. IEEE transactions on pattern analysis and machine intelligence, 35(1), 221-231.
[79] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 28.
[80] Spanhol, F. A., Oliveira, L. S., Petitjean, C., & Heutte, L. (2016, July). Breast cancer histopathological image classification using convolutional neural networks. In 2016 international joint conference on neural networks (IJCNN) (pp. 2560-2567). IEEE.
[81] Spanhol, F. A., Oliveira, L. S., Petitjean, C., & Heutte, L. (2015). A dataset for breast cancer histopathological image classification. Ieee transactions on biomedical engineering, 63(7), 1455-1462.
[82] Cireşan, D., Meier, U., Masci, J., & Schmidhuber, J. (2012). Multi-column deep neural network for traffic sign classification. Neural networks, 32, 333-338.
[83] Lin, H., Jia, J., Qiu, J., Zhang, Y., Shen, G., Xie, L, & Chua, T. S. (2017). Detecting stress based on social interactions in social networks. IEEE Transactions on Knowledge and Data Engineering, 29(9), 1820-1833.
[84] Abdel-Hamid, O., Mohamed, A. R., Jiang, H., & Penn, G. (2012, March). Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition. In 2012 IEEE international conference on Acoustics, speech and signal processing (ICASSP) (pp. 4277-4280). IEEE.
[85] Mohamed, A. R., Dahl, G. E., & Hinton, G. (2011). Acoustic modeling using deep belief networks. IEEE transactions on audio, speech, and language processing, 20(1), 14-22.