Models of Convolutional Neural Network

0. Abstract

In recent years, deep learning has been used in image classification, object detection, pose estimation, text detection and recognition, action recognition and scene labeling. For image classification, among different types of models, Convolutional neural networks(CNN) have been demonstrated high performance. In this paper, I will introduce about CNN and some popular models of CNN, AlexNet, VGG and ResNet, which has developed in order. Experiments are based on ImageNet dataset.

1. Introduction

Computer vision has become increasingly important and effective in recent years due to its wide-ranging applications in areas as diverse as smart surveillance and monitoring, health and medicine, sports and recreation, robotics, drones, and self-driving cars. Visual recognition tasks, such as image classification, localization, and detection, are the core building blocks of many of these applications, and recent developments in Convolutional Neural Networks (CNNs) have led to outstanding performance in these state-of-the-art visual recognition tasks and systems. As a result, CNNs now form the crux of deep learning algorithms in computer vision.[1]

CNN is useful in many applications, especially in image related tasks. Applications of CNN include image classification, image semantic segmentation, 2 object detection in images, etc. Image classification plays an important role in computer vision. We will focus on image classification (or categorization) in this paper. In image categorization, every image has a major object which occupies a large portion of the image. An image is classified into one of the classes based on the identity of its main object, e.g., dog, airplane, bird, etc. One key ingredient of deep learning in image classification is the use of Convolutional architectures

Convolutional neural network design inspiration comes from the mammalian visual system structure. Convolutional neural network is first introduced by LeCun in [2]. Since 2006, many methods have been developed to overcome the difficulties encountered in training deep neural networks. Krizhevsky propose a classic CNN architecture AlexNet[3] and show significant improvement upon previous methods on the image classification task. With the success of AlexNet[3], several works are proposed to improve its performance. ZFNet[4], VGGNet[5], GoogleNet[6] and ResNet[7] are proposed. In this work, the architecture and training methods of Convolutional neural networks, specifically AlexNet[3], VGGNet[5] and ResNet[7] will be introduced.

2. Convolutional Neural Network

2.1. Basic CNN components

Convolutional neural network layer types mainly include three types, namely Convolutional layer, pooling layer, and fully-connected layer. When these layers are stacked, a CNN architecture has been formed. A simplified CNN architecture for MNIST classification is illustrated in Figure 1.

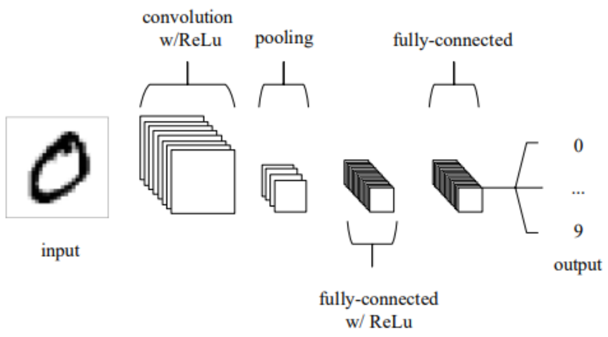


Figure 1 A simple CNN architecture, comprised of just five layers. [9]

2.1.1. Convolution layer

Convolutional layer is the core part of the Convolutional neural network, which has local connections and weights of shared characteristics. The aim of Convolutional layer is to learn feature representations of the inputs. As shown in above, Convolutional layer is consisting of several feature maps. Each neuron of the same feature map is used to extract local characteristics of different positions in the former layer, but for single neurons, its extraction is local characteristics of same positions in former different feature map. In order to obtain a new feature, the input feature maps are first convolved with a learned kernel and then the results are passed into a nonlinear activation function. We will get different feature maps by applying different kernels.[8] The rectified linear unit (commonly shortened to ReLu) aims to apply an 'elementwise' activation function such as sigmoid to the output of the activation produced by the previous layer.

Convolutional layers are also able to significantly reduce the complexity of the model through the optimization of its output. These are optimized through three hyperparameters, the depth, the stride and setting zero-padding. The depth of the output volume produced by the convolutional layers can be manually set through the number of neurons within the layer to the same region of the input. We are also able to define the stride in which we set the depth around the spatial dimensionality of the input in order to place the receptive field. For example, if we were to set a stride as 1, then we would have a heavily overlapped receptive field producing extremely large activations. Alternatively, setting the stride to a greater number will reduce the amount of overlapping and produce an output of lower spatial dimensions. Zero-padding is the simple process of padding the border of the input and is an effective method to give further control as to the dimensionality of the output volumes.

It is important to understand that through using these techniques, we will alter the spatial dimensionality of the convolutional layers output. To calculate this, you can make use of the following formula[9]:

$$\frac{(\mathrm{V} - \mathrm{R}) + 2\mathrm{Z}}{S + 1}$$

Where V represents the input volume size (height×width×depth), R represents the receptive field size, Z is the amount of zero padding set and S referring to the stride. If the calculated result from this equation is not equal to a whole integer, then the stride has been incorrectly set, as the neurons will be unable to fit neatly across the given input.

2.1.2. Pooling layer

Pooling layers aim to gradually reduce the dimensionality of the representation, and thus further reduce the number of parameters and the computational complexity of the model.[9] It is usually placed between two Convolutional layers. The size of feature maps in pooling layer is determined according to the moving step of kernels. The typical pooling operations are average pooling and max pooling. We can extract the high-level characteristics of inputs by stacking several Convolutional layer and pooling layer.[8]

2.1.3. Fully-connected layer

The fully-connected layers will then perform the same duties found in standard ANNs (Figure 2) and attempt to produce class scores from the activations, to be used for classification. In general, the classifier of Convolutional neural network is one or more fully-connected layers. They take all neurons in the previous layer and connect them to every single neuron of current layer. There is no spatial information preserved in fully-connected layers. The last fully-connected layer is followed by an output layer. For classification tasks, softmax regression is commonly used because of it generating a well-performed probability distribution of the outputs. Another commonly used method is SVM, which can be combined with CNNs to solve different classification tasks.[8]

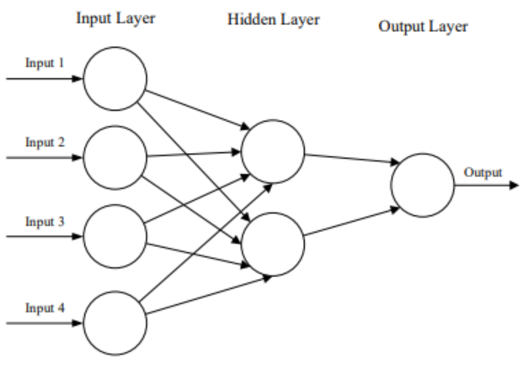


Figure 2 A simple three-layered feedforward neural network, comprised of an input layer, a hidden layer, and an output layer. [9]

2.2. Models (Architectures)

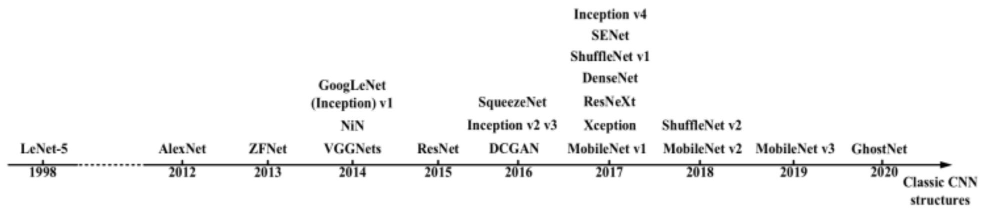


Figure 3 Part of classic CNN models. [9]

Compared with other methods, CNNs can achieve better classification accuracy on large scale datasets due to their capability of joint feature and classifier learning. The breakthrough of large-scale image classification comes in 2012. Krizhevsky et al. [3] develop the AlexNet and achieve the best performance in ILSVRC (ImageNet Large Scale Visual Recognition Competition) 2012. After the success of AlexNet, several works have made significant improvements in classification accuracy by either reducing filter size or expanding the network depth. Part of well-known models can be seen in Figure 3. We will discuss about AlexNet and two more significant models, VGGNet and ResNet.

2.2.1. AlexNet

The architecture of our AlexNet is summarized in Figure 4. It contains eight learned layers - five convolutional and three fully-connected. The output of the last fully-connected layer is sent to a 1000-way softmax layer which corresponds to 1000 class labels in the ImageNet dataset.

Before AlexNet, sigmoid and tanh were usually used as activation function, but AlexNet used Rectified Linear Units (ReLUs) activation function which are non-saturating nonlinearity. The formula of ReLU is $f(x) = \max(0, x)$. It avoids vanishing gradients for positive values and more computationally efficient to compute and has better convergence performance than sigmoid and tanh.

In AlexNet, 1.2 million training parameters are too big to fit into the NVIDIA GTX 580 GPU with 3GB of memory. Therefore, they spread the network across two GTX 580 GPUs. It is just for memory limitation. AlexNet used overlapping max pooling of size 3x3 with stride 2. This scheme reduced the top-1 and top-5 error rates by 0.4% and 0.3%. Local Response Normalization (LRN) is used in AlexNet to help with generalization. LRN reduces the top-1 and top-5 error rates by 1.4% and 1.2%. Nowadays, batch normalization is used instead of LRN. A regularization technique called Dropout was used in AlexNet. It randomly set the output of each hidden neuron to zero with the probability of 0.5. Dropout reduces complex co-adaptations of neurons since a neuron cannot rely on the presence of particular other neurons.

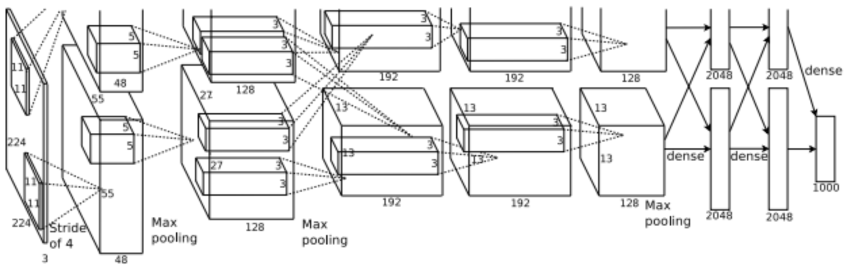


Figure 4 An illustration of the architecture of AlexNet. [3]

2.2.2. VGGNet

K. Simonyan et al. [5] address another important aspect of AlexNet architecture design – its depth. To this end, they fix other parameters of the architecture, and steadily increase the depth of the network by adding more convolutional layers, which is feasible due to the use of very small (3 × 3) convolution filters in all layers.

This showed experimentally that the parallel assignment of these small-size filters could produce the same influence as the large-size filters. In other words, these small-size filters made the receptive field similarly efficient to the large-size filters (7×7 and 5×5). By decreasing the number of parameters, an extra advantage of reducing computational complication was achieved by using small-size filters. These outcomes established a novel research trend for working with small-size filters in CNN.

In general, VGG obtained significant results for localization problems and image classification. While it did not achieve first place in the 2014-ILSVRC competition, it acquired a reputation due to its enlarged depth, homogenous topology, and simplicity. However, VGG's computational cost was excessive due to its utilization of around 140 million parameters, which represented its main shortcoming. Figure 5 shows the structure of the network.

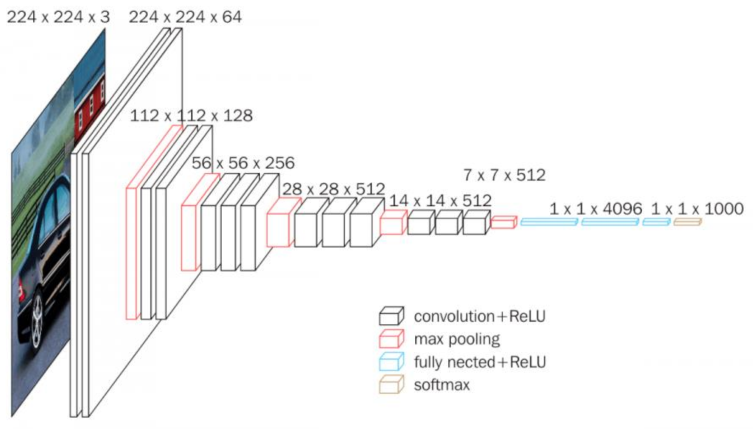


Figure 5 An architecture of VGGNet. [11]

2.2.3. ResNet

For conventional deep learning networks, they usually have conv layers then fully connected (FC) layers for classification task like AlexNet[3], ZFNet[4] and VGGNet[5], without any skip, shortcut connection, we call them plain networks here. When the plain network is deeper (layers are increased), the problem of vanishing/exploding gradients occurs. To solve the problem of vanishing/exploding gradients, a skip/shortcut connection is added to add the input x to the output after few weight layers as Figure 6. Even if there is vanishing gradient for the weight layers, we always still have the identity x to transfer back to earlier layers.

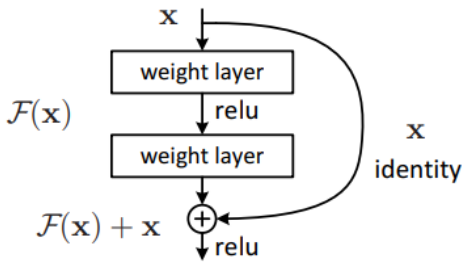


Figure 6 Residual learning: a building block. [7]

Figure 7 shows the ResNet architecture[7]. In Figure 7, the VGG-19 [5] (bottom) is a state-of-the-art approach in ILSVRC 2014. And 34-layer plain network (middle) is treated as the deeper network of VGG-19. The Resnet, 34-layer residual network (top) is the plain one with addition of skip connection. There are 3 types of skip connections when the input dimensions are smaller than the output dimensions. In addition, Batch Normalization is used for each conv.

Skip connection enables to have deeper network and finally ResNet becomes the Winner of ILSVRC 2015 in image classification, detection, and localization, as well as Winner of MS COCO 2015 detection, and segmentation. By increasing the network depth to 152 layers, 5.71% top-5 error rate is obtained which is much better than VGG-16[5] and GoogleNet[6].

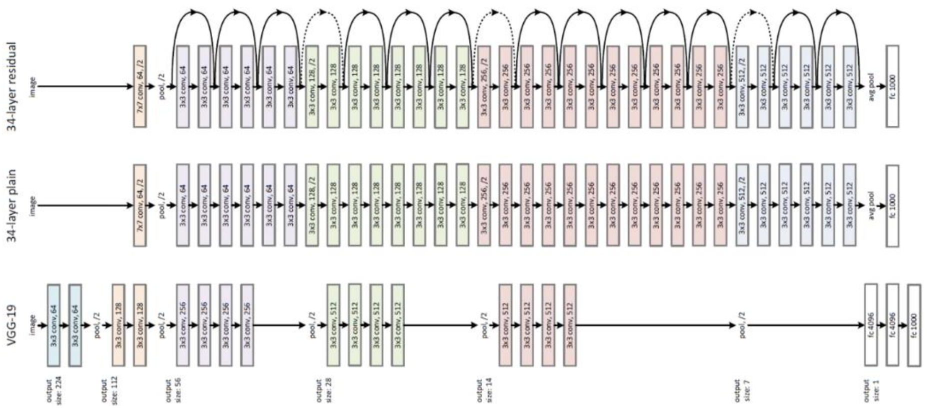


Figure 7 Example network architecture for ImageNet. **Below**: the VGG-19 model, **Middle**: a plain network with 34 layers, **Above**: a residual network with 34 layers. The dotted shortcuts increase dimensions. [7]

2.3. Dataset – ImageNet

ImageNet is a dataset of over 15 million labeled high-resolution images belonging to roughly 22,000 categories. The images were collected from the web and labeled by human labelers using Amazon's Mechanical Turk crowd-sourcing tool. Starting in 2010, as part of the Pascal Visual Object Challenge, an annual competition called the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) has been held. ILSVRC uses a subset of ImageNet with roughly 1000 images in each of 1000 categories. In all, there are roughly 1.2 million training images, 50,000 validation images, and 150,000 testing images.

Models were evaluated on the ILSVRC classification dataset [12] that consists of 1000 classes. The models are trained on the 1.3 million training images and evaluated on the 50k validation images. Final result was obtained on the 100k test images, reported by the test server. On ImageNet, it is customary to report two error rates: top-1 and top-5, where the top-5 error rate is the fraction of test images for which the correct label is not among the five labels considered most probable by the model.

2.3.1. Training details

ImageNet consists of variable-resolution images, while AlexNet method requires a constant input dimensionality. Therefore, for AlexNet[3], the images are down-sampled to a fixed resolution of 256 × 256. Given a rectangular image, the image was first rescaled such that the shorter side was of length 256, and then cropped out the central 256×256 patch from the resulting image. AlexNet model was trained by using stochastic gradient descent with a batch size of 128 examples, momentum of 0.9, and weight decay of 0.0005. We trained the network for roughly 90 cycles through the training set of 1.2 million images.

The images for VGG[5] and ResNet[7] are down-sampled to 224 × 224 for scale augmentation. A 224×224 crop is randomly sampled from an image or its horizontal flip, with the per-pixel mean subtracted. For VGG model, the batch size was set to 256, momentum to 0.9. The training was regularized by weight decay (the L2 penalty multiplier) and dropout regularization for the first two fully-connected layers (dropout ratio set to 0.5). The learning rate was initially set to 10^{-2}, and then decreased by a factor of 10 when the validation set accuracy stopped improving. In total, the learning rate was decreased 3 times, and the learning was stopped after 370K iterations (74 epochs).

For ResNet model, batch normalization (BN) [13] was adopted right after each convolution and before activation. SGD was used with a mini-batch size of 256. The learning rate starts from 0.1 and is divided by 10 when the error plateaus, and the models are trained for up to 60×10^4 iterations. A weight decay is 0.0001 and a momentum is 0.9. Dropout was not used.

3. Result

3.1. AlexNet

Table 1 Comparison of error rates on ILSVRC-2012 validation and test sets. [3]

Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SIFT + FVs [14]	—	—	26.2%
1 CNN	40.7%	18.2%	—
5 CNNs	38.1%	16.4%	**16.4%**
1 CNN*	39.0%	16.6%	—
7 CNNs*	36.7%	15.4%	**15.3%**

The results are summarized in Table 1. The CNN means AlexNet model. It achieves a top-5 error rate of 18.2%. Averaging the predictions of five similar CNNs gives an error rate of 16.4%. Training one CNN, with an extra sixth convolutional layer over the last pooling layer, to classify the entire ImageNet Fall 2011 release (15M images, 22K categories), and then "fine-tuning" it on ILSVRC-2012 gives an error rate of 16.6%. Averaging the predictions of two CNNs that were pre-trained on the entire Fall 2011 release with the aforementioned five CNNs gives an error rate of 15.3%. The second-best contest entry achieved an error rate of 26.2% with an approach that averages the predictions of several classifiers trained on FVs computed from different types of densely-sampled features [14].

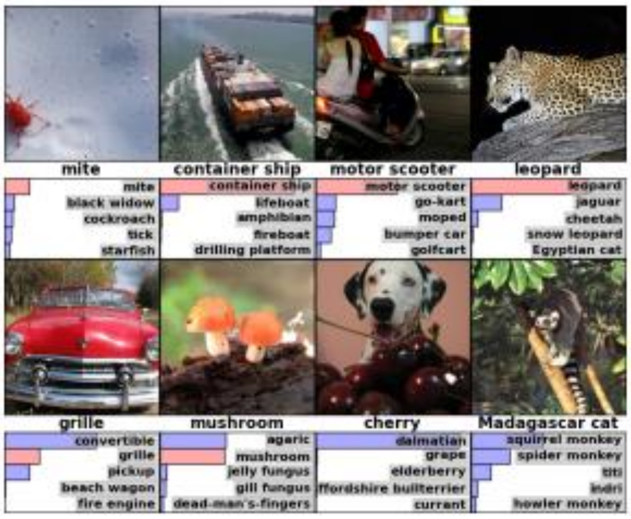


Figure 8 Eight ILSVRC-2010 test images and the five labels considered most probable by AlexNet. [3]

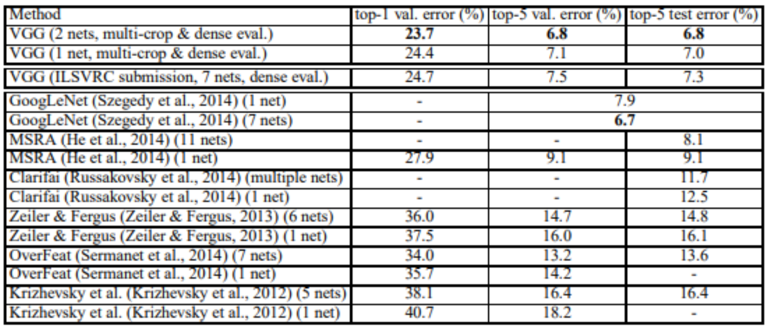
In the left panel of Figure 8, we qualitatively assess what the network has learned by computing its top-5 predictions on eight test images. Notice that even off-center objects, such as the mite in the top-left, can be recognized by the net. Most of the top-5 labels appear reasonable. For example, only other types of cats are considered plausible labels for the leopard. In some cases (grille, cherry) there is genuine ambiguity about the intended focus of the photograph.

3.2. VGG

As Seen in Table 2, VGG significantly outperform the previous generation of models, which achieved the best results in the ILSVRC-2012 and ILSVRC-2013 competitions. The result is also competitive with respect to the classification task winner (GoogleNet with 6.7% error) and substantially outperforms the ILSVRC-2013 winning submission Clarifai, which achieved 11.2% with outside training data and 11.7% without it. This is remarkable, considering that the best result is achieved by combining just two models – significantly less than used in most ILSVRC submissions.

In terms of the single-net performance, VGG achieves the best result (7.0% test error), outperforming a single GoogleNet by 0.9%.

Table 2 Comparison with the state of the art in ILSVRC classification. [5]



Method	top-1 val. error (%)	top-5 val. error (%)	top-5 test error (%)
VGG (2 nets, multi-crop & dense eval.)	**23.7**	**6.8**	**6.8**
VGG (1 net, multi-crop & dense eval.)	24.4	7.1	7.0
VGG (ILSVRC submission, 7 nets, dense eval.)	24.7	7.5	7.3
GoogLeNet (Szegedy et al., 2014) (1 net)	-	7.9	
GoogLeNet (Szegedy et al., 2014) (7 nets)	-	**6.7**	
MSRA (He et al., 2014) (11 nets)	-	-	8.1
MSRA (He et al., 2014) (1 net)	27.9	9.1	9.1
Clarifai (Russakovsky et al., 2014) (multiple nets)	-	-	11.7
Clarifai (Russakovsky et al., 2014) (1 net)	-	-	12.5
Zeiler & Fergus (Zeiler & Fergus, 2013) (6 nets)	36.0	14.7	14.8
Zeiler & Fergus (Zeiler & Fergus, 2013) (1 net)	37.5	16.0	16.1
OverFeat (Sermanet et al., 2014) (7 nets)	34.0	13.2	13.6
OverFeat (Sermanet et al., 2014) (1 net)	35.7	14.2	-
Krizhevsky et al. (Krizhevsky et al., 2012) (5 nets)	38.1	16.4	16.4
Krizhevsky et al. (Krizhevsky et al., 2012) (1 net)	40.7	18.2	-

3.3. ResNet

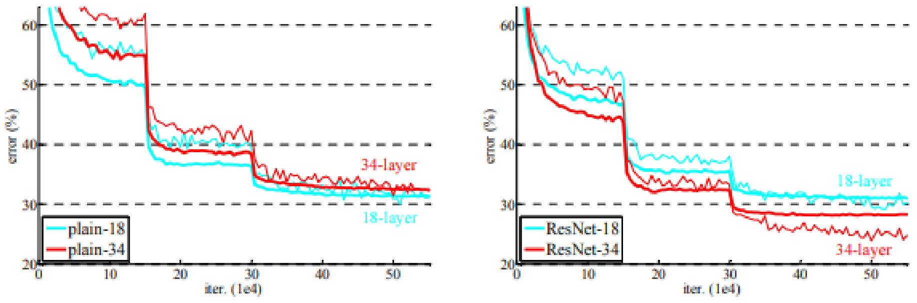
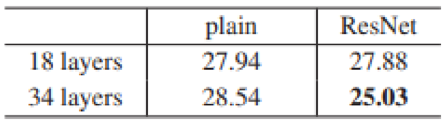


Figure 9 Training procedures. Thin curves denote training error, and bold curves denote validation error of the center crops. **Left**: plain networks of 18 and 34 layers. **Right**: ResNets of 18 and 34 layers. [7]

Table 3 Top-1 error on validation. [7]



	plain	ResNet
18 layers	27.94	27.88
34 layers	28.54	**25.03**

The 34-layer plain net is in Figure 7 (middle), and the 18-layer plain net is of a similar form. First evaluating 18-layer and 34-layer plain nets, the results in Table 3 show that the deeper 34-layer plain net has higher validation error than the shallower 18-layer plain net. On the other hand, 34-layer ResNet has lower validation error than the 18-layer ResNet.

There are three major observations from Table 3 and Figure 9. First, the situation is reversed with residual learning – the 34-layer ResNet is better than the 18-layer ResNet (by 2.8%). This indicates that the degradation problem is well addressed in this setting. Second, compared to its plain counterpart, the 34-layer ResNet reduces the top-1 error by 3.5% (Table 3), resulting from the successfully reduced training error (Figure 9 right vs. left). This comparison verifies the effectiveness of residual learning on extremely deep systems. Last, the 18-layer plain/residual nets are comparably accurate (Table 3), but the 18-layer ResNet converges faster (Figure 9 right vs. left). When the net is “not overly deep” (18 layers here), the current SGD solver is still able to find good solutions to the plain net. In this case, the ResNet eases the optimization by providing faster convergence at the early stage.

In Table 4, three options are compared: (A) zero-padding shortcuts are used for increasing dimensions, and all shortcuts are parameter free; (B) projection shortcuts are used for increasing dimensions, and other shortcuts are identity; and (C) all shortcuts are projections.

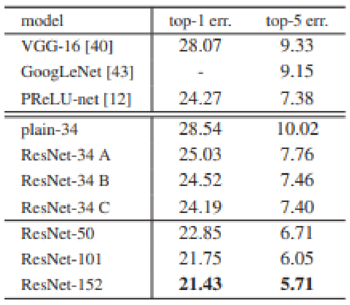


Table 4 Error rates on validation. ResNet-50/101/152 are of option B that only uses projections for increasing dimensions. [7]

model	top-1 err.	top-5 err.
VGG-16 [40]	28.07	9.33
GoogLeNet [43]	-	9.15
PReLU-net [12]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**

Table 4 shows that all three options are considerably better than the plain counterpart. B is slightly better than A. This is because the zero-padded dimensions in A indeed have no residual learning. C is marginally better than B, and they attribute this to the extra parameters introduced by many (thirteen) projection shortcuts. But the small differences among A/B/C indicate that projection shortcuts are not essential for addressing the degradation problem.

4. Conclusion

In this paper, I introduced about Convolutional Neural Network for image classification. Convolutional Neural Networks differ to other forms of Artificial Neural Network in that instead of focusing on the entirety of the problem domain, knowledge about the specific type of input is exploited. This in turn allows for a much simpler network architecture to be set up[9].

Thus, different methods of CNN, AlexNet, VGG and ResNet, were proposed. Each model outperformed previous works and has great performance on each year ILSVRC challenge. Those methods were used in many works for many years, and currently, ResNet is most widely used. Even currently, more efficient methods of image classification are still developing.

5. References

[1] Khan, Salman, et al. "A guide to convolutional neural networks for computer vision." Synthesis Lectures on Computer Vision 8.1 (2018): 1-207.

[2] LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.

[3] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems 25 (2012).

[4] Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." European conference on computer vision. Springer, Cham, 2014.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in ICLR, 2015.

[6] Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[7] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[8] Guo, Tianmei, et al. "Simple convolutional neural network on image classification." 2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA). IEEE, 2017.

[9] O'Shea, Keiron, and Ryan Nash. "An introduction to convolutional neural networks." arXiv preprint arXiv:1511.08458 (2015).

[10] Li, Zewen, et al. "A survey of convolutional neural networks: analysis, applications, and prospects." IEEE Transactions on Neural Networks and Learning Systems (2021).

[11] VGG16 – Convolutional Network for Classification and Detection.
URL https://neurohive.io/en/popular-networks/vgg16/

[12] Russakovsky, Olga, et al. "Imagenet large scale visual recognition challenge." International journal of computer vision 115.3 (2015): 211-252.

[13] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." International conference on machine learning. PMLR, 2015.

[14] ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC 2012).
URL http://www.image-net.org/challenges/LSVRC/2012/.