A Comparative Study of AlexNet and ResNet

David Felipe Alvear Goyes

September 15, 2024

1 Introduction

Deep learning has had great development in the past decade in multiple fields such as image recognition, image segmentation, natural language processing, and others. In particular, the task of developing a model to classify images has required a lot of effort to address this problem, from traditional machine learning algorithms to deep learning models. Convolutional neural networks have been at the forefront of this challenge giving groundbreaking results. Especially, AlexNet and ResNet are two convolutional network architectures that provided important contributions to the field of image recognition.

2 AlexNet

AlexNet paper created a turning point for deep learning models, being one of the first to formulate a deep learning model that achieved remarkable results in the ImageNet dataset. Object recognition task exposes a high complexity, and given the lack of sufficient data the authors offer a solution using convolutional neural networks which is the largest network defined to date, optimized by a GPU 2D convolution implementation for CNN training.

The CNN proposed by [1] consists of 5 convolutional layers and three fully connected layers followed by a 1000 softmax activation. Where the authors introduced the use of a Non-saturating nonlinearity activation function ReLu. This function leads a faster training than other options like tanh and generates positive results in timing efficiency for training models in larger datasets. ReLu is applied to the output of every convolutional layer and fully connected layer. Local response normalization is included in layer 1 and 2 between the convolutional layer and the activation function, this aid in generalization in training, and the effect of this configuration is to reduce the error rate. Max-pooling layers are introduced after the convolutional layers 1-2, and 5 are adding complexity to avoid overfitting [1].

Given that the network contains 60 million parameters, the authors discuss two main methods for reducing overfitting. The first method is to use data augmentation to enlarge the dataset with label-preserving transformations. The proposed method includes translations, and transformations from extracted patches of the original downsampled images, and also discuss the transformation of altering channel RGB intensities using principal component analysis. Data augmentation is designed to be computationally free consisting of performing the modifications and transformations on the CPU, while the model is training in previous batches. Dropout regularization is discussed in [1], which consists in setting to zero the output of hidden layers in the network that have a probability equal to 0.5. The consequence is that dropped-out layers don't contribute to the forward pass and backpropagation. This method reduces complex co-adaptations of neurons, leading to learning robust features. However, this implementation requires double of iterations to converge.

For training the proposed deep neural network the authors used 2 GPUs spreading the network between the devices, where half of the kernels are on each GPU, and the network has communication connections in certain layers. The use of GPUs to train the network leads to the reduction of error rates and will require less time to train. Additionally, it used stochastic gradient descent with a batch size of 128 for the optimization, 0.9 momentum, and weight decay of 0.0005. The weights are initialized first and the biases are set to 1 in certain layers to help the model to accelerate the learning at early stages, any positive input in ReLu activations will trigger the function. The learning rate dynamically changes reducing the current value by 10 when the validation error stops improving. Finally, the model took 5 to 6 days to train using 2 GPUs [1].

The proposed model achieves top-1 and top-5 test errors of 37.5% and 17.0% respectively in the ILSVRC-2010 context. Due to the network division between the GPUs being observable in the specialization of each unit, the network learned frequency and orientation-selective kernels [1].

Concluding, AlexNet generated ground braking results using a deep convolutional neural network in a challenging dataset like ImageNet. The depth is important for the results. However, at that time the results were far from human-vision performance.

3 ResNet

This paper addresses the problem of vanishing gradients in deep neural networks due to accuracy degradation. The ResNet paper developed the following contributions to gain accuracy increasing the depth of the model [2].

The authors propose a deep residual learning framework reformulating the layers as residual functions. Traditional deep-learning networks use underlying mapping functions to approximate complicated functions. Similarly, a stack of layers also can fit a residual function to approximate complicated functions [2]. This new definition addresses the problem of degradation of accuracy due to vanishing gradients. The addition of layers that learn a residual function is designed with the use of identity mappings so that a deeper model with this additional layer can have a training error no greater than its simpler counterpart model.

The formulation of the building block is defined as follows: $y = F(x, \{W\}) + x$, where x is the input, y is the output of the layers, and $F(x, \{W\})$ is the residual mapping to learn. The component of this block is a shortcut connection by adding the input to the residual function[2]. In particular, when the dimension of the input and the residual function does not match exist two approaches: the first one is to perform a zero padding or use a linear projection Ws in the shortcut $y = F(x, \{W\}) + W_s x$. In both the proposed method solves the degradation issue, however, the first method has the advantage that in the new function to be fit we only add a parameter not increasing computation complexity. The formulation of the residual function works when the stack of the layers is greater than one [2].

To test the reformulation of residual blocks they implemented plain networks of two ImageNet models, one based in VGG-19 with 19 layers (19.6 billion FLOPs), and a plain network of 34 layers of 3.6 billion FLOPs consisting of 33 convolutional layers with 3x3 filter.Downsampling is performed by convolutional layers with the stride of 2, followed by a global average pooling with a 1000 fully connected layer and softmax activation. The comparison of the performance of these networks evidences the problem of degradation. The designed residual network is based on the 34 plain networks, including the residual block every two layers with the following approach, when the input and output of the layers share the same dimension they implement an identity mapping function. Due to the mismatch in the dimensions, it is implemented an identity mapping with zero-padding to increase dimension and not add an extra parameter. The authors suggest a projection shortcut adding an extra parameter to overcome the dimensional mismatch issue, however, it is not essential for solving the degradation problem. With the resulting residual network with 34 layers and 3.6 billion FLOPs, they implement batch normalization between the convolution and activation. The training of the networks is done from scratch using SGD optimization with a mini-batch of 256, $60 * 10^4$ iterations, weight decay of 0.001, and momentum of 0.9 [2].

For the experiments with the plain and residual networks, the study used the ImageNet 2012 dataset with 1000 classes training with 1.8 million training images, 50k for validation, and 100k test images. For the evaluation, they focused on top-1 and top-5 errors.

After the implementation of the plain networks, they evidenced that the degradation problem exists in the experiment, the 34-layer network has worse validation and training error compared with an 18-layer plain network. Additionally, the results exposed exponentially low convergence rates in plain networks. On the other hand, the results of the 34-layer residual network improved an 18-layer ResNet with 2.8%, this means that it achieved the goal of accuracy gain with the increase of depth, also they evidenced a lower training error and generalizable validation data. Furthermore, the 18-layer ResNet resulted in a faster convergence than the counterpart 18-layer plain network, due to the learning of residual functions.

The authors of [2] modified the network to achieve a very deep neural network using a bottleneck configuration with a parameter-free identity shortcut. This modification leads to the creation of 101 and 152 layers of ResNet with 11.3 billion FLOPs achieving better performance than 34-layer networks, not observing degradation problems and increasing gain accuracy.

Finally, the authors stacked multiple layers to create deeper networks to test the CIFAR-10 dataset, resulting in no optimization difficulties, but having a higher training, and testing error that shallowed networks [2]. The performance of this experiment is due to overfitting due to the small dataset and the complexity of the network. But it still has a good generalization and performance having a small training and test error.

4 Comparision

AlexNet and ResNet represent two turning points in image recognition. The contributions of the two papers paved the way for modern deep-learning architectures and influenced the development of new models. Notice that the architectures were developed at different times, so more than a comparison is a discussion of the contributions to the field.

AlexNet was published in 2012, winning the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012. One of the main contributions was outperforming traditional machine learning methods for image recognition. AlexNet developed a simpler architecture of 8 layers while introducing key advancements in deep learning, popularizing convolutional layers, ReLu activation function, data augmentation, and regularization techniques. Similarly, ResNet contributions managed the problem of vanishing gradients and created the groundbreaking contribution of training very deep learning models as never seen before. ResNet introduced the concept of the residual block, which allowed the network to be deeper without affecting by the degradation problem. In terms of performance, AlexNet provided a new level of image recognition with its contributions, winning the ILSVRC-2012 with a top-5 error rate of 15.3%, revealing the capability of GPUs for training models, and providing a new state-of-the-art. ResNet developed a 152-layer architecture with a top-5 error rate of 3.57% winning ILSVRC 2015. This performance is due to the ability to solve the vanishing gradient problem.

Concluding, AlexNet and ResNet showed that deep neural networks achieve state-of-the-art image classification tasks. Both inspired the development of different architectures, and their contributions paved the path to contemporary deep-learning models. AlexNet introduced key innovations such as convolutional layers and ReLu activations, while ResNet introduced residual blocks as a solution to performance degradation, leading to the building of deep-learning architectures.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 1097–1105.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

Binary Cross Entropy Loss:

$-\frac{1}{n}\sum_{i=1}^{n}[y_i\log(\hat{y}_i)+(1-y_i)\log(1-\hat{y}_i)]$
where:

n is the number of samples y_i is the true label of the i-th sample (either 0 or 1) $\hat{y}_i$ is the predicted probability of the i-th sample belonging to class 1 The binary cross entropy loss is used for binary classification problems where the output of the model is a probability value between 0 and 1.

Binary Cross Entropy with Logits Loss:
$-\frac{1}{n}\sum_{i=1}^{n}[y_i \cdot \text{logsigmoid}(\hat{y}_i) + (1 - y_i) \cdot \text{logsigmoid}(-\hat{y}_i)]$
where:

n is the number of samples y_i is the true label of the i-th sample (either 0 or 1) $\hat{y}_i$ is the predicted value (logit) of the i-th sample $\text{logsigmoid}(x) = \log(\frac{1}{1+\exp(-x)})$