A Modified PyramidNet for CIFAR-10 Image Classification

Jeremy R. Carleton
Department of Electrical and Computer Engineering
Texas A&M University
jcarleton@tamu.edu

Abstract

A form of residual convolutional neural network known as the PyramidNet which gradually increases the number of filters layer-by-layer rather than abruptly increasing it at a few specific layers was able to achieve state-of-the-art performance on the CIFAR-10 benchmark when it was first introduced in 2016. With improvements to the training process, it has continued to perform competitively over the years. In this paper, we discuss how the performance of the PyramidNet can be improved by changing the activation function and applying various techniques which improve generalization ability. We present a PyramidNet with under 11 million parameters which achieves 95.35% accuracy on the CIFAR-10 test data.

1 Introduction

The CIFAR-10 dataset [Krizhevsky, 2009] is a popular dataset for training convolutional neural networks, in part because it is possible for networks with very modest sizes to achieve decent accuracy. However, to achieve state-of-the-art accuracy one typically resorts to much deeper networks with many more feature maps which presents the issue of how best to ensure that useful representations are able to move through the full network. While the introduction of residual connections in He et al. [2015b] brought a significant improvement on this front, the issue of how best to transition from the very small number of feature maps output by the first convolutional layer to the very large number of feature maps used at the output remained an open question. In this project, we evaluate an alternative to traditional choices for when to expand the number of feature maps. We also consider a non-traditional activation function, the Gaussian Error Linear Unit, and employ significant data augmentation alongside dropout to enhance generalizability. Out of the two network configurations that we consider, the better one achieves 95.35% test accuracy when trained on 100% of the training data.

2 Design features

2.1 PyramidNet architecture

The core design for the network presented in this paper is based on the PyramidNet architecture introduced by Han et al. [2016] which gradually increases the number of feature maps block-by-block rather than only on the blocks which perform downsampling, as is seen in ResNet. The layout of a pyramidNet is defined by the number of groups of blocks, the number of blocks per group N_i, and the widening factor α. This is used to determine the number of feature maps processed by the 3x3 convolution inside each bottleneck block, and the number of input and output feature maps to the block is four times that amount. When building the model that they test on CIFAR-10, the creators of PyramidNet consider only the case where there are three groups of blocks, and use a hyperparameter called “depth" to control the number of blocks per group (which is the same for all groups). If D is

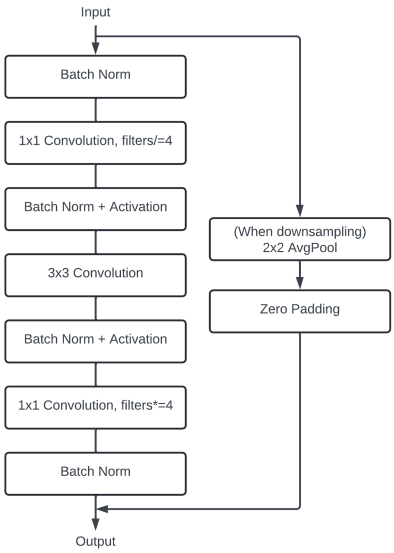

Figure 1: Layout of the bottleneck block used in PyramidNet

the value of depth, then

$$N_i = (D-2)/\lambda,\ \forall i \in \{1,2,3\}$$

where $\lambda = 6$ if standard blocks are used and $\lambda = 9$ if bottleneck blocks are used. Given the depth and the widening factor, the number of additional feature maps f processed by the 3x3 convolution in each consecutive block is calculated as

$$\text{FM}_{\text{addrate}} = \frac{\alpha}{\sum_{i=1}^{n} N_i} = \frac{\alpha\lambda}{3(D-2)}$$

During the design phase, two different models were trained to explore the trade-off between additional layers and additional feature maps. The first used $D = 110$ and $\alpha = 200$, while the second used $D = 150$ and $\alpha = 170$. Due to GPU memory constraints, the number of parameters was kept below 11 million for both models. Table 1 gives further details for both of these network configurations.

The structure of the residual blocks used in the network (which matches those of Han et al. [2016]) is presented in Figure 1. The main pathway features convolutional layers interleaved with batch norm and activation layers much like in the original ResNet [He et al., 2015b], except there is no activation layer before the first convolution and an additional batch norm layer is inserted before combining the pathways. On the other hand, the residual pathway features significant differences from that of the original ResNet. First, downsampling is performed using a 2D average pooling layer with a kernel size and stride of 2 in both dimensions as opposed to a convolutional layer. Second, zero padding is used to make the number of feature maps at the end of the residual pathway match the number of feature maps at the end of the main pathway. Han et al. [2016] show that this is superior to using a 1x1 convolution to perform the projection because it avoids the use of learnable parameters which can overfit to training data or impede the flow of gradient information through the residual pathway. Intuitively, zero-padding preserves all of the feature maps entering the block and hence makes it easiest for the model to learn an identity mapping for those feature maps.

The main body of our network is preceded by a 3x3 convolutional layer which produces 16 output feature maps followed by a batch norm layer. At the end of the network is a final batch norm and activation layer, a dropout layer, a global average pooling layer with a kernel size of 8, and a fully connected layer to produce the output logits. The number of feature maps which are processed by the global average pooling layer for both the network configurations tested is given in Table 1.

2.2 Gaussian error linear unit activation function

The activation function chosen for the final network is the Gaussian Error Linear Unit (GELU) introduced by Hendrycks and Gimpel [2016]. This activation function was considered because it

Table 1: Comparison of network configurations

Configuration	# Parameters	# Convolutions	# Output Feature Maps
$D = 110$, $\alpha = 200$	10,862,034	109	888
$D = 150$, $\alpha = 170$	10,754,874	145	760

has displayed superior performance to both the Rectified Linear Unit (ReLU) used in the original PyramidNet and the Exponential Linear Unit (ELU) on the CIFAR 10 and 100 datasets [Hendrycks and Gimpel, 2016]. Before using GELU as the activation function, the Parametric ReLU activation function [He et al., 2015a] was considered, but it failed to boost performance over the generic ReLU.

The GELU activation function can be viewed as a smoothed version of the ReLU which deviates most significantly from ReLU in the vicinity of $x = 0$, although at the point $x = 0$ they match exactly. The GELU function is given by

$$\text{GELU}(x) = x \cdot \Phi(x)$$

where $\Phi(x)$ is the cumulative distribution function of the standard normal distribution. This function has the advantage of not immediately flattening out when x is negative, which helps avoid the problem of too many feature map pixels losing influence on the output due to their outputs being pushed into the negative range.

2.3 Data augmentation

To enhance the generalization capabilities of our network, we apply three kinds of augmentations to the training images: color augmentation, random cropping, and horizontal reflection. Color augmentation is done by jittering the brightness, contrast, and saturation of the image; we do not modify the hue directly since the objects represented in the data typically only appear in a small variety of colors which are likely already well-represented in the training data, and we expect that training images which recolor the entire image in an outlandish way are unlikely to help the network generalize to the test images. The torchvision transform ColorJitter is used to apply this augmentation. Random cropping is applied by padding the edges of the image with four pixels and then choosing a pixel out of the upper-left 9x9 square of the padded image to be the upper-left corner of the cropped image. Finally, horizontal reflection is applied probabilistically such that the image has an equal probability of ending up in either orientation.

The level of data augmentation increases through three phases during training. For the first 8 epochs, no data augmentation is applied so the model sees the same images repeatedly. The purpose of this phase is for the model to quickly learn useful representations that can be refined in the later phases; the short duration of the phase and the presence of the dropout layer ensures that the model will not overfit immediately. For the next 52 epochs, the images are augmented with color augmentation or random cropping with equal probability, and images have a 50% chance to be flipped horizontally. For the rest of the epochs, the same augmentations are used but the color augmentation and random cropping are both applied to every image. Note that as more data augmentation is applied, the time to process an epoch increases slightly; this is why a range of times is given in Table 2.

2.4 Dropout

While the authors of PyramidNet do not use dropout in their original network, it presents the possibility to further improve the generalization ability of the network. Hence, we apply dropout to the output feature maps right before the global average pool is applied. This is achieved using a Dropout2d layer from Pytorch which zeroes out an entire feature map when it is dropped. Since we use bottleneck blocks, there is a relatively large number of feature maps entering the global average pool layer, so intuitively the network should be capable of achieving good accuracy even with several of those feature maps removed. The dropout probability we use is 20%.

3 Training configuration

Following Han et al. [2016], we use stochastic gradient descent with Nesterov momentum for our optimizer. When training the two potential networks for the purpose of comparison, we trained for

Table 2: Training statistics

Configuration	Epoch Time Range (s)	Best Val. Acc.	Epoch for Best Checkpoint
$D = 110$, $\alpha = 200$	126 - 141	95.22%	170
$D = 150$, $\alpha = 170$	148 - 162	95.06%	200

200 epochs with a learning rate of 0.1 that was scaled by a factor of 0.1 every 60 epochs. The batch size and weight decay used were 64 and 0.0001, respectively. We started phase one augmentation after 8 epochs and phase two augmentation after 60 epochs. Checkpoints were saved every 10 epochs, and the highest validation accuracy was taken from those checkpoints. In this case, the validation accuracy was calculated using 5000 held-out images from the training data. Table 2 gives the training results for each of the two main network configurations tested. After identifying the epoch where the validation accuracy was maximized, both models were then trained for the same number of epochs on the full training data (i.e. including the 5000 images used as validation data in the previous runs) which increased the epoch time by up to 15 seconds.

4 Results

Looking at Table 2, it appears that the network configuration with more feature maps has an edge over the network configuration which is deeper. However, we found that after retraining on the full training data, the test accuracy of 95.35% achieved by the deeper network was higher than the test accuracy of 95.29% achieved by the shallower network. Overall, the difference in performance is not substantial, which suggests that configurations of PyramidNet using a similar number of parameters are likely to have a similar performance. It is worth noting, however, that shallower networks have the advantage of training faster.

5 Conclusion

In this report, we have presented a model which achieves 95.35% accuracy on the 10000 test images from the CIFAR-10 dataset. There are still some areas where our network could be improved; for example, we noticed that it has significantly more difficulty differentiating images of cats and dogs than any other pair of classes. Future work towards solving that issue, along with further hyperparameter tuning, could make the network even more accurate.

References

D. Han, J. Kim, and J. Kim. Deep pyramidal residual networks. *CoRR*, abs/1610.02915, 2016. URL `http://arxiv.org/abs/1610.02915`.

K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, Los Alamitos, CA, USA, dec 2015a. IEEE Computer Society. doi: 10.1109/ICCV.2015.123. URL `https://doi.ieeecomputersociety.org/10.1109/ICCV.2015.123`.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015b. URL `http://arxiv.org/abs/1512.03385`.

D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus). 2016. doi: https://doi.org/10.48550/arXiv.1606.08415.

A. Krizhevsky. Learning multiple layers of features from tiny images. pages 32–33, 2009. URL `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`.