Incorporating Handcrafted Features in Wide Residual Networks

Pratik Vaishnavi
pvaishnavi@cs.stonybrook.edu
Stony Brook University

Leena Shekhar
lshekhar@cs.stonybrook.edu
Stony Brook University

Abstract

Deep residual networks have emerged as a family of extremely deep neural network architectures that show high performance on image classification while being more difficult to train. In this paper we work with an architecture that uses residual building blocks with identity mappings as the skip connections and activation function as "pre-activation" of the weight layers. Similar work has been done in recent past [7], but nobody has incorporated handcrafted features along with identity mappings. This motivates us to propose a new residual unit, and we will demonstrate its performance on CIFAR-10 dataset. Our code is available on github at https://github.com/pratik18v/CSE-527-Project.

1. Introduction

Deep residual networks can be scaled up to many layers and still have higher performance. The superiority of deep networks has been proved in several recent works. A residual block with identity mapping can be represented as follows:

$$x_{l+1} = x_l + \mathcal{F}(x_l, \mathcal{W}_l)$$

where x_{l+1} and x_l are input and output of the l-th unit respectively, $\mathcal{F}$ is a residual function, and $\mathcal{W}$ are parameters of the block [7]. The latest residual networks had a huge success winning ImageNet and COCO 2015 competition and achieving state-of-the-art in object classification. These networks are superior in terms of generalization and display nice convergence behavior. Our goal in this paper is to explore a much richer architecture of ResNet blocks and examine its performance on CIFAR-10 data set. We argue that features such as dense SIFT and HoG can be added to these blocks to increase the performance of the network for classification tasks.

2. Dataset

The CIFAR-10 dataset has 60000 images, 50000 training images data and 10000 test images. Entire dataset is divided into 10 mutually exclusive classes with 6000 images for each class. Each image is 32×32 in dimension, is coloured, and belongs to one of the 10 classes. This dataset is divided into 6 batches- 5 training batches and 1 test batch. Each batch has 10000 images; test batch has 1000 randomly selected images for each class. The dataset has following 10 classes: Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, Truck [6].

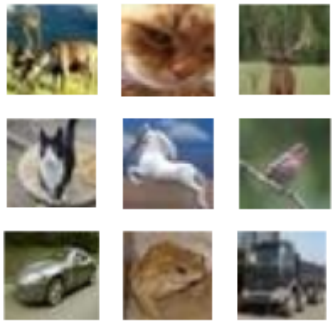


Figure 1: Visualizing input to the network- CIFAR-10 images after mean/std normalization.

3. Related Work

1. Deep residual network architecture described in [5] have proved the importance of skip connections in deep residual networks. It showed how each residual unit performs the following computation:

$$y_l = h(x_l) + \mathcal{F}(x_l, \mathcal{W}_{\updownarrow})$$

$$x_{l+1} = f(y_l)$$

where x_l and x_{l+1} are input and output of the l-th residual unit, $\mathcal{F}$ is a residual function, and $\mathcal{W}$ is a set of weights. The function f is ReLU and h is an identity mapping. The above equations can be combined and written as follows

$$x_{l+1} = x_l + \mathcal{F}(x_l, \mathcal{W}_{\updownarrow})$$

$$x_L = x_l + \sum_{i=l}^{L-1} \mathcal{F}(x_i, \mathcal{W}_{\rangle})$$

$$\frac{\partial \mathcal{E}}{\partial x_l} = \frac{\partial \mathcal{E}}{\partial x_L}\left(1 + \frac{\partial}{\partial x_l}\sum_{i=l}^{L-1} \mathcal{F}(x_i, \mathcal{W}_{\rangle})\right)$$

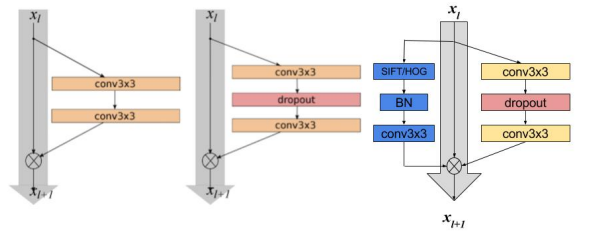
The gradient can be decomposed into two additive terms: a term that propagates information without concerning any weight layers and a term that passes through the weight layers. This makes sure that the gradient of a layer does not vanish even when the weights are small. This is the reason why we have identity skip channel in our architecture.

2. Wide residual networks [7] discussed the importance of residual blocks in such networks and proposed a novel widened architecture that allows improved performance. It shows network performance for different combinations of depth and width of the residual block. Thin and deep residual networks with small kernels have sequential structure, which is against the nature of GPU computations. On the other hand wide networks help effectively balance computations in a much more optimal way and thus are more efficient. Wide networks with only 16 layers can significantly outperform 1000-layer deep networks on CIFAR.

4. Residual Network

Even with less number of layers wide residual networks have proved to perform significantly better than deep residual networks. Networks with only 16 layers can significantly outperform 1000 layer deep networks on CIFAR, as well as that 50 layer outperform 152 layer on ImageNet [7]. This proves the fact that structure of residual blocks is of utmost importance in a ResNet. Our work builds on top of the previous work done by [7, 5] and tries to answer the question of how wide deep residual networks can be modified to improve performance on object classification. Structure of the wide residual network proposed by us is as follows:

1. Residual block consists of two consecutive 3×3 convolutions with batch normalization and ReLU preceding convolution. Each convolution layer has 3×3 filter size as small filers have been shown to be very effective.
2. An initial convolution layer is followed by three groups of residual blocks, followed by average pooling and a fully connected layer for final classification.
3. The size of initial convolution layer is fixed while the widening factor k scales the width of the residual blocks in the other three convolution layers.



(a) basic-wide (b) Wide with dropout (c) Proposed wide-dropout with sift features

Figure 2: Proposed Residual Block. Experimental results on wide residual networks show that when dropout is inserted between the two convolutional layers network shows improved performance.

4. In the SIFT model, input to the input layer is of dimension (3+128)×32×32 where first 3 indices represent the three color channels for the image and the rest 128 indices represent dense SIFT feature for the image.
5. In the HoG model, input to the input layer is of dimension (3+1)×32×32 where first 3 indices represent the three color channels for the image and the rest 128 indices represent dense SIFT feature for the image.

group name	output size	block type = B(3,3)
convolution 1	32×32	[3×3, 16]
convolution 2	32×32	[3×3, 16×K]×N, [3×3, 16×K]×N
convolution 3	16×16	[3×3, 32×K]×N, [3×3, 32×K]×N
convolution 4	8×8	[3×3, 64×K]×N, [3×3, 64×K]×N
avg-pool	1×1	[8×8]

Table 1: Structure of wide residual network. *K* determines the width of the network and *N* is the number of blocks in convolution groups. We have used a ResNet block of type B(3,3) in our architecture similar to [7], where B(M) denote residual block structure and M is a list with the kernel sizes of the convolutional layers in a block.

5. Implementation Details

We have made several modifications in the original architecture implementation [3] to be able to incorporate sift features. The main idea is to have a channel parallel to residual blocks and skip connection, which carries either SIFT (referred to as model 1) or HoG (referred to as model 2) features. This extra channel will converge with the other two channels at the element-wise addition point in the architecture. We have 3 residual blocks in our network, each block has 18 residual units, each unit has 2 covolutional layer and width of 1. We decided to not use filters larger than 3×3 as small filters have shown to be very effective in the past. Thus different parameters used for training are as

follows, $d = 3$, $N = 18$, $l = 2$, and $k = 1$. We used decaying learning rate, .01 till 120-th iteration after which it was .001. For all our evaluations we have referred to model with added SIFT channel as model 1 and model with added HoG channel as model 2.

5.1. Tools Used

All the code written is in Python3.5 and utilizes the various open-source libraries like: NumPy, Scikit, OpenCV-Python etc.

1. OpenCV is an open-source BSD-licensed library that includes various computer vision algorithms. In our project we have used its Python interface to calculate SIFT descriptors for the images.
2. Scikit-image is an image processing toolbox for SciPy. We used this library to calculate HoG descriptors for the images.
3. Lasagne is a scientific open source lightweight library to build and train neural networks in Theano. We used this framework to implement and train our ResNet model. Lasagne has many pre-defined neural network layers and also packages that enable us to run models on GPUs.

5.2. Input

The network takes input in the form 3×32×32 images of the CIFAR-10 dataset, where 3 refers to the RGB channels and 32 refers to the image dimension. For all our experiments we used mean/std normalization as the preprocessing step. In model 1 we compute the dense SIFT descriptors for each input image, which come out to be of size 128×32×32 where 128 represent length of SIFT features vector for each pixel. These 128 features are concatenated at the end of the 3 RGB channels to get the input of size 131×32×32. In model 2 we compute HoG features vector for the image, which is of size 1×32×32. Later in the network we slice the input along the first dimension to extract RGB and SIFT/HoG channels in separate variables.

5.3. Handcrafted Features

- SIFT descriptors calculated for every pixel of the image is referred to as dense SIFT features. This dense feature extraction is shown to have provide higher object category recognition accuracy than key point based feature extraction. A 16x16 neighbourhood around the key point is taken which is then divided into 16 sub-blocks of 4x4 size. For each sub-block, 8 bin orientation histogram is created. Since for each pixel of the image 128 bin values are available, so for an image 128×32×32 bin values are present [2].
- HoG descriptors are calculated for every image and fed to the network in a separate channel. We calculated HoG by taking 8 orientations, (2,2) pixels per cell and (1,1) cells per block. We get 1 descriptor for every pixel so for an image this is 1×32×32 in dimension [1].

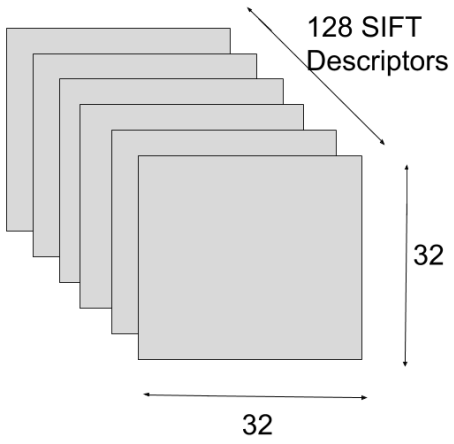


Figure 3: SIFT descriptors as input to the network.

5.4. The Slice Layer

- In model 1 input to our network is of size 131×32×32. We now insert a split layer after the input layer to split this input along the first dimension to separate its RGB and SIFT channels. The RGB channels are passed through the Residual Blocks whereas the SIFT features are passed through Batch normalization and a convolutional layer along the SIFT channel. The output of the series of Residual Blocks, input through the skip connection and the SIFT features of the input are all added element-by-element to produce the output of our Residual Unit.
- In model 2 input to our network is of size 4×32×32. We now insert a split layer after the input layer to split this input along the first dimension, similar to the SIFT channel, and separate the RGB and HoG channels. Rest of the steps are similar to the above mentioned steps model 1.

5.5. Element-wise addition

As the element-wise operation requires size of the output of feature channel to be same as the output of the first residual block (as we have added the hand-crafted features only to the first residual block), we set the number of filters in the convolutional layer in the feature channel in such a way that matrix dimensions are compatible at time of element-wise addition.

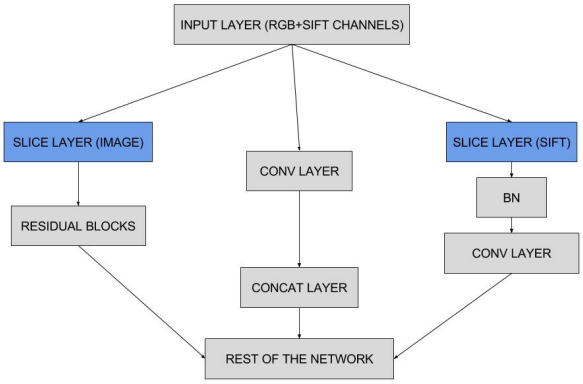


Figure 4: Flow in the first Residual block

6. Experimental Results

All the experiments have been done using the architecture shown in Figure 2, taking $K = 1$ and $N = 18$, on the CIFAR-10 dataset. The graphs show training and validation loss vs number of iterations for the network. Time per iteration increases as we increase the number of features, therefore is maximum for SIFT model and minimum for the baseline model. Ad mentioned for all our evaluations we have referred to model with added SIFT channel as model 1 and model with added HoG channel as model 2.

7. Observations

1. Model 1 performed considerably worse than model 2 on 1/10 of the dataset and for 100 iterations. We think this was due to the fact that SIFT introduced too many features into the network (128 features for every pixel), as compared to HoG (1 feature per pixel) and caused the model to over fit. As can be seen from Table 2 both the models performed worse than the baseline model, model with only identity skip channel and no hand crafted features.
2. Performance of model 1 became comparable to the other two models as training data size was increased. This observation is in unison with the fact that when more features are added, more data is required to train the model without causing over-fitting. This can be verified from Table 3.
3. Effect of adding batch normalization and using smaller learning rate resulted in both the models to exhibit a better convergence behavior as can be seen from Figure 7.
4. As training data size was increased both the model exhibited better convergence behavior as can be seen in Figure 8.
5. Model 2 performed better than the baseline model by giving an improvement of 5 % as an be seen from Table 4.

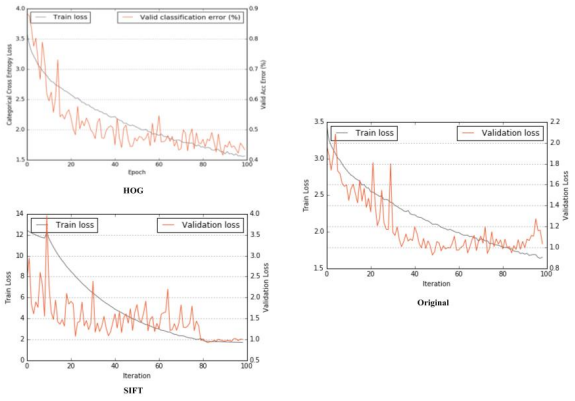


Figure 5: Training curves for the SIFT, HoG, and Baseline models on $1/10$ of CIFAR-10 dataset for 100 iterations. Wide ResNet depth = 18 and width = 1.

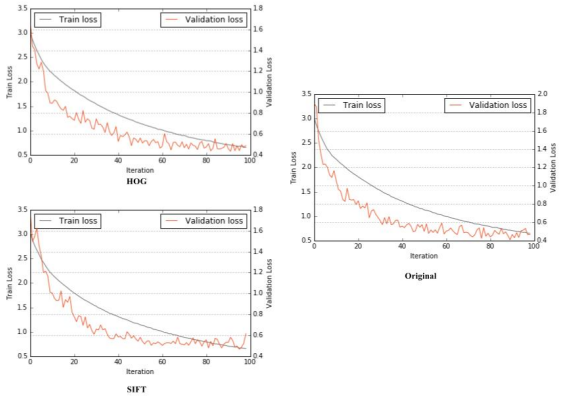


Figure 6: Training curves for the SIFT, HoG, and Baseline models on $1/2$ of CIFAR-10 dataset for 100 iterations. Wide ResNet depth = 18 and width = 1.

8. Conclusion

A new architecture was presented for object classification and a comparative study was done to evaluate its performance with the baseline model presented in [7]. Our proposed model performed better than the baseline model on CIFAR-10 dataset. We wanted to run the proposed architecture for larger width and depth values to check the veracity of our hypothesis, but due to machine constraints we were not able to do so. Though we believe that our model has the

potential to perform better than the final model, WRN-28-10, proposed in [7].

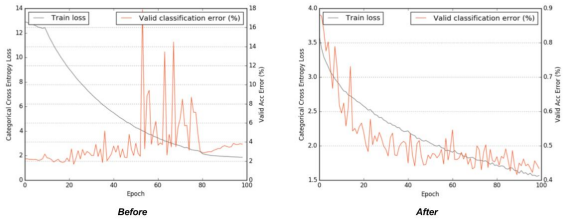


Figure 7: As seen from the graph, the validation accuracy flickers a lot between 50 and 80 epochs, the reason being higher learning rate between these epochs. We decreased the learning rate to get a smoother curve for validation error. Also, the performance was pretty bad for which we planned on adding a batch-normalization layer in the SIFT channel. This ensured all our data getting added in the element-wise addition layer were in the same range numerically. Having manipulated SIFT features this way, we then added a convolutional layer after the batch normalization layer. The output of this layer was given to the element-wise addition layer. Wide ResNet depth = 18 and width = 1.

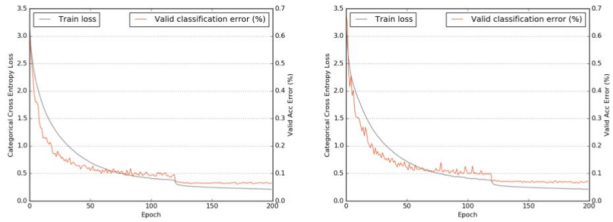


Figure 8: Training curves for the HoG and Baseline models on full CIFAR-10 dataset for 200 iterations. Wide ResNet depth = 18 and width = 1.

Model	Val acc (%)	Test acc (%)
Original	78.21	76.45
SIFT	58.42	55.12
HOG	71.58	69.44

Table 2: Results on 1/10 dataset, trained for 100 iterations

Model	Val acc (%)	Test acc (%)
Original	87.54	87.75
SIFT	86.60	86.04
HOG	86.71	85.93

Table 3: Results on half dataset, trained for 100 iterations

Model	Val acc (%)	Test acc (%)
Original	93.55	93.40
HOG	93.91	93.54

Table 4: Results on full dataset, trained for 200 iterations

References

[1] Opencv.

[2] Scikit hog.

[3] FlorianMuellerklein. Identity mapping resnet lasagne.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. *arXiv preprint arXiv:1603.05027*, 2016.

[6] A. Krizhevsky. Cifar dataset.

[7] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146v1*, 2016.