CIS 680 Homework #2

Oleh Rybkin
University of Pennsylvania

Introduction

The task of the homework was to further experiment with neural networks. I completed the task based on the provided example code.

Task 1

In this task I experimented with data augmentation techniques and their influence on the network performance. The training accuracies are on the Fig. 1 and the test accuracies are in the Tab. 1.

Task #	1.1	1.2	1.3	1.4
Test accuracy, %	48	51	63	64

Table 1: Task 1. Test accuracy.

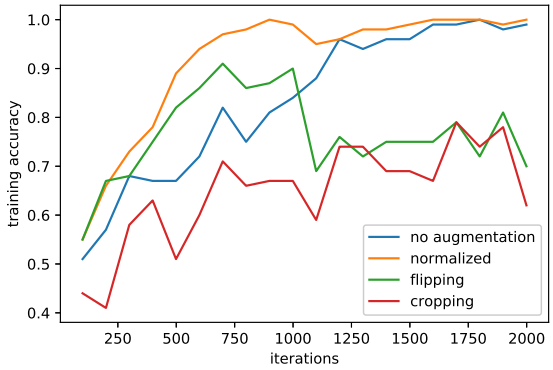


Figure 1: Task 1. Training accuracy progress.

In the initial experiment the network is able to memorize almost all training data, but still achieves significantly higher than chance test accuracy. When adding image normalization, the test accuracy marginally increases. We expect this, as normalized input helps keep the gradients at constant level, so that an image doesn't cause big shift in weights just because of e.g. a change in brightness. Another perspective is that normalizing removes a degree of freedom from the image which is not relevant for the classification task.

The test accuracy further increases when images are randomly flipped and cropped. The network is forced now to learn some kind of partial invariance to those transformations. As the classification labels are indeed invariant to them, the network learns a more useful representation and performs better on previously unseen data. Moreover, the level of overfitting decreases drastically, perhaps because it is now harder to explain the variations in the training data, and the network does not have the time to memorize it all.

Task 2

In this task I implemented a network without fully connected layers. I followed the paper [4] and used the ConvPoll-CNN-C model from it, which I provide in the Tab. 2. The network essentially delays the execution of the final pooling until after fully connected layers. The FC layers thus become convolutional 1×1 layers, and can be viewed as FC layers operating on each neuron separately. T The training accuracy of this and derived networks is on the Fig. 4 and the test accuracy is in the Tab. 3.

One interesting observation I had during development is that the network did not achieve satisfactory performance when using 64 filters for the first 3 layers. It also overfit-

Input: 32×32 RGB image
3×3 conv. 96 ReLU 3×3 conv. 96 ReLU 3×3 conv. 96 ReLU
3×3 max-pooling stride 2
3×3 conv. 192 ReLU 3×3 conv. 192 ReLU 3×3 conv. 192 ReLU
3×3 max-pooling stride 2
3×3 conv. 192 ReLU
1×1 conv. 192 ReLU
1×1 conv. 10 ReLU
global averaging over 6×6 spatial dimensions
10 softmax with cross-entropy

Table 2: Task 2. Network architecture.

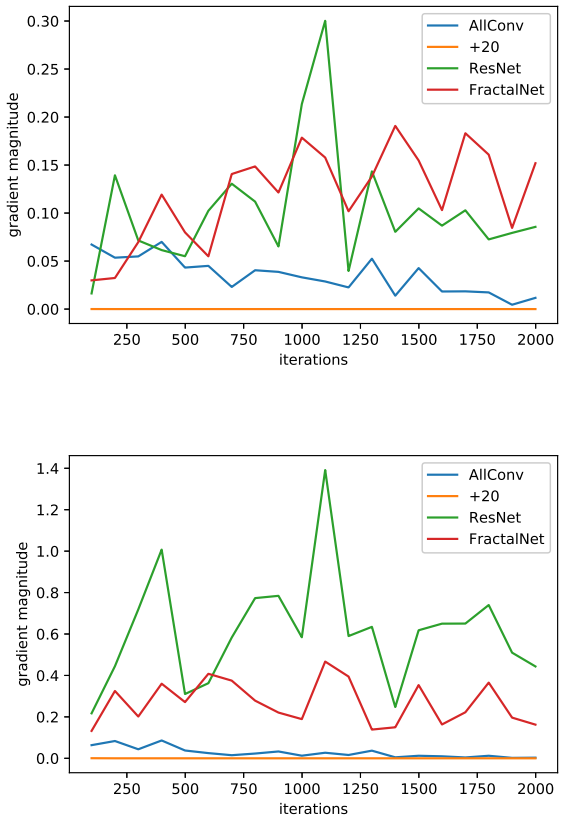


Figure 2: Task 2. Gradient values. Top: first layer. Bottom: last layer.

ted strongly, which is not the case for the final architecture. This goes against traditional understanding that the network overfits more when it has more trainable parameters.

An alternative for this approach might have been a more involved data augmentation, such as random rotations, changing contrast and brightness, or adding noise to the image. I might also have explored only-convolutional AlexNet-like architecture.

Next I add 20 convolutional levels into the network and remove all BatchNorm layers. We observe that the network is no longer able to learn anything. The gradient values also become much smaller.

I note that while gradient values are small, the network does not seem to converge to any solution. In fact, both training and test accuracies fluctuate randomly over the

Task #	2.1	2.3	2.4 Res	2.4 Fractal
Test accuracy, %	73	8	51	54

Table 3: Task 2. Test accuracy.

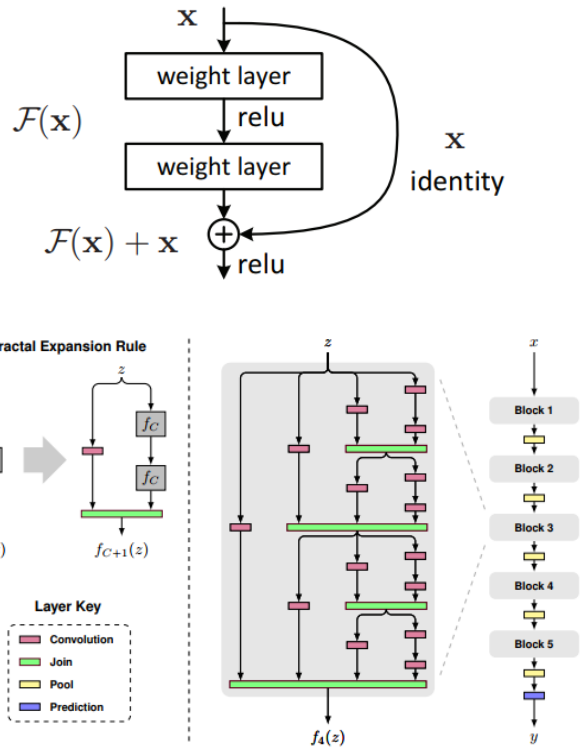


Figure 3: Networks with skip-connections. Top: ResNet block. Bottom: FractalNet block.

epochs and do not display any structure. The gradient of the last layer is also very small, which means that the problem is not in activation function or scaling of gradients as they go through the network. It follows that the actual problem is that the network cannot figure out the true gradient direction and is stuck in a space where any change does not help to improve the accuracy. I conclude that the gradient vanishing is in fact a symptom rather than an actual problem

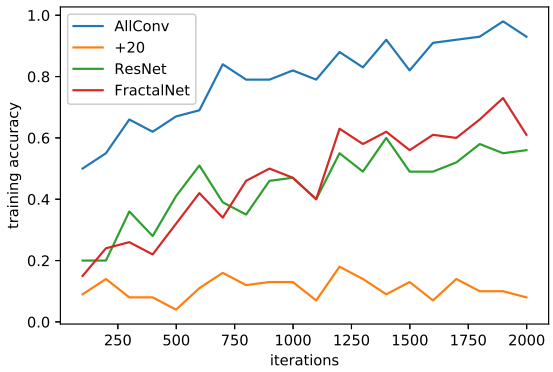


Figure 4: Task 2. Training accuracy progress.

in this case.

I explore two solutions two this problem based on the idea of skip-connections: a ResNet-like ([2]) and a FractalNet-like ([3]) architecture. In the first case I divide the 20 additional layers into 10 Residual blocks (see Fig. 3, top). In the second case I divide the layers into 5 FractalNet blocks, where one block is as indicated on the Fig. 3, bottom. The only difference from the picture is that I use blocks with three different pathways instead of four. Note that the rightmost way in the network, when the five blocks are stacked together, creates the original 20-layer structure. As opposed to the original work, I do not use dropout neither of individual neurons nor of entire layers. The network, however, is still able to train, both in case of ResNet and FractalNet, and achieve approximately 50 percent test accuracy. The intuition as to why skip-connections help training very deep network is well described in the referenced papers. In short, there exists a short path through the network, and when restricted to that path the network is able to learn something. By extension, even the layers in longer paths can learn as they are now provided with meaningful gradients.

Note that adding skip-connections from the first to the 21st layer would technically be a solution for this task. Yet another solution might be to to delete the ReLU layers in the additional layers, effectively making them one layer. Neither of it would however fully explore the potential of the very deep layers

Task 3

In this task I experimented with adversarial images and their generation.

I train the networks from scratch instead of loading the weights. The images are standardized before going through the process and restored to original scale after. I also removed BatchNorm layers from the first network. The behaviour with the batch norm layer was as follows: if the *is_train* parameter is set to *False*, the computed gradient values are NaN only; if the parameter is set to *Train*, the computed gradient values look normal, but when applied to the image, the confidence scores remain close to 0.1. Without BatchNorm, gradients applied to the image quickly drive it to be wrongly classified. This behaviour change was thoroughly tested with minimal examples.

Network #	From task 1	From task 2
Test accuracy, %	15	15

Table 4: Task 3. Test accuracy.

As in the CIFAR-10 dataset images are very small, it is hard to generate an adversarial image that would be indistinguishable from the original. Essentially, it is not possible to distribute change over the image in an unnoticable way. However, some of the generated images look very similar to the original, and all original images are still easily correctly classified by a human.

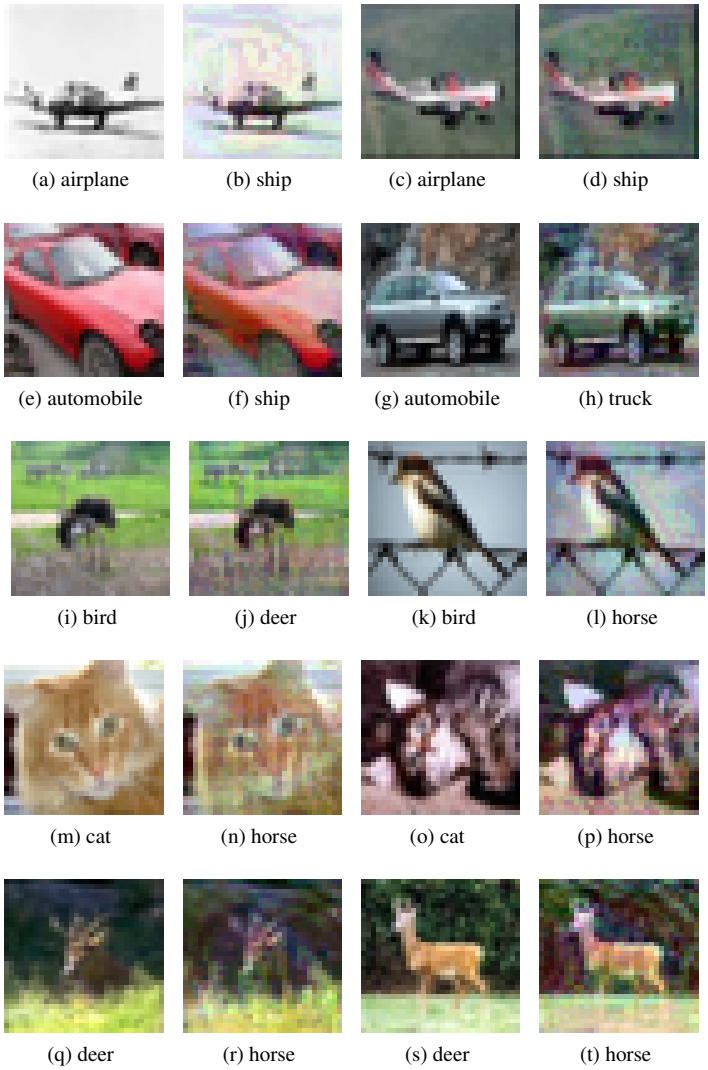
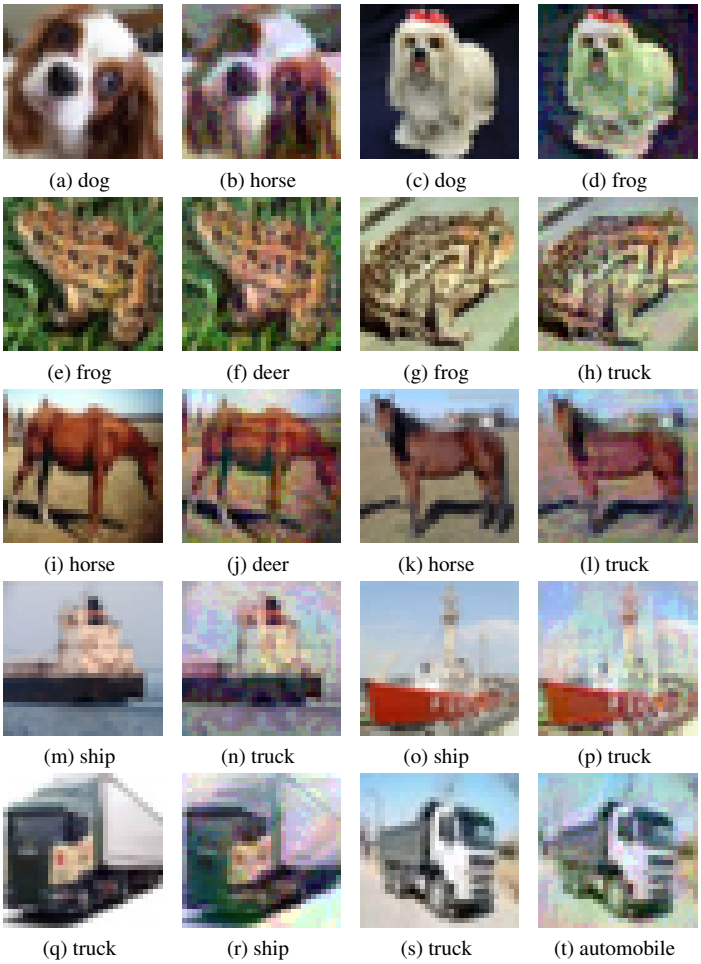


Figure 5: Adversarial images. The original image is always to the left of the perturbed one.

The test accuracy of another network on the adversarial images is very low (see Tab. 4). This agrees with the observations from the literature (e.g. [1]). The accuracy of the network with which the adversarial images were generated, when retrained once again on the same data, is also low (but not zero, as with the original network).

Closing remarks

I gained practical exprerience in creating neural architectures, combating vanishing gradient problem and generating



(a) dog (b) horse (c) dog (d) frog

(e) frog (f) deer (g) frog (h) truck

(i) horse (j) deer (k) horse (l) truck

(m) ship (n) truck (o) ship (p) truck

(q) truck (r) ship (s) truck (t) automobile

Figure 6: More adversarial images

adversarial images.

I would like to thank Nikolaos Kolotouros and Karl Pertsch with whom I discussed various appoaches for completing the assignment.

References

[1] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. 12 2014.

[2] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[3] G. Larsson, M. Maire, and G. Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. *CoRR*, abs/1605.07648, 2016.

[4] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller. Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806, 2014.