BLG 506E COMPUTER VISION

ASSIGNMENT 5
Convolutional Nets: The Last Stand

Student No: 504221511

Name & Surname: Enis Teper

1. Download assignment 2 from Stanford's CS231n:
http://cs231n.github.io/assignments2019/assignment2/
You will be following ConvolutionalNetworks.ipnyb notebook. You will implement several layers of convolutional networks, then use these layers to train on the CIFAR-10 dataset to get state-of-the- art results. We are surrounded.

Dataset and assignment is downloaded to drive.

2. Implement conv_forward_naive function in cs231n/layers.py . Test your implementation on the notebook. For our people.

1. Stride and padding values are extracted from input.
2. All shapes from input and weight are extracted to use them at calculating output sizes.
3. Output sizes are controlled since they have to be integers.
4. Output array is generated as zeros with respect to output shapes.
5. Zero padding is done to surrounding of input.
6. Finally kernel is moved around image with for loops to obtain final output.

3. To have a better understanding of convolution operation you are presented some image processing operations in the notebook, just examine and run it. The aid is coming.

With moving kernel over image, grayscale image and image of edges are obtained. The representation of images are shown at Figure 1.

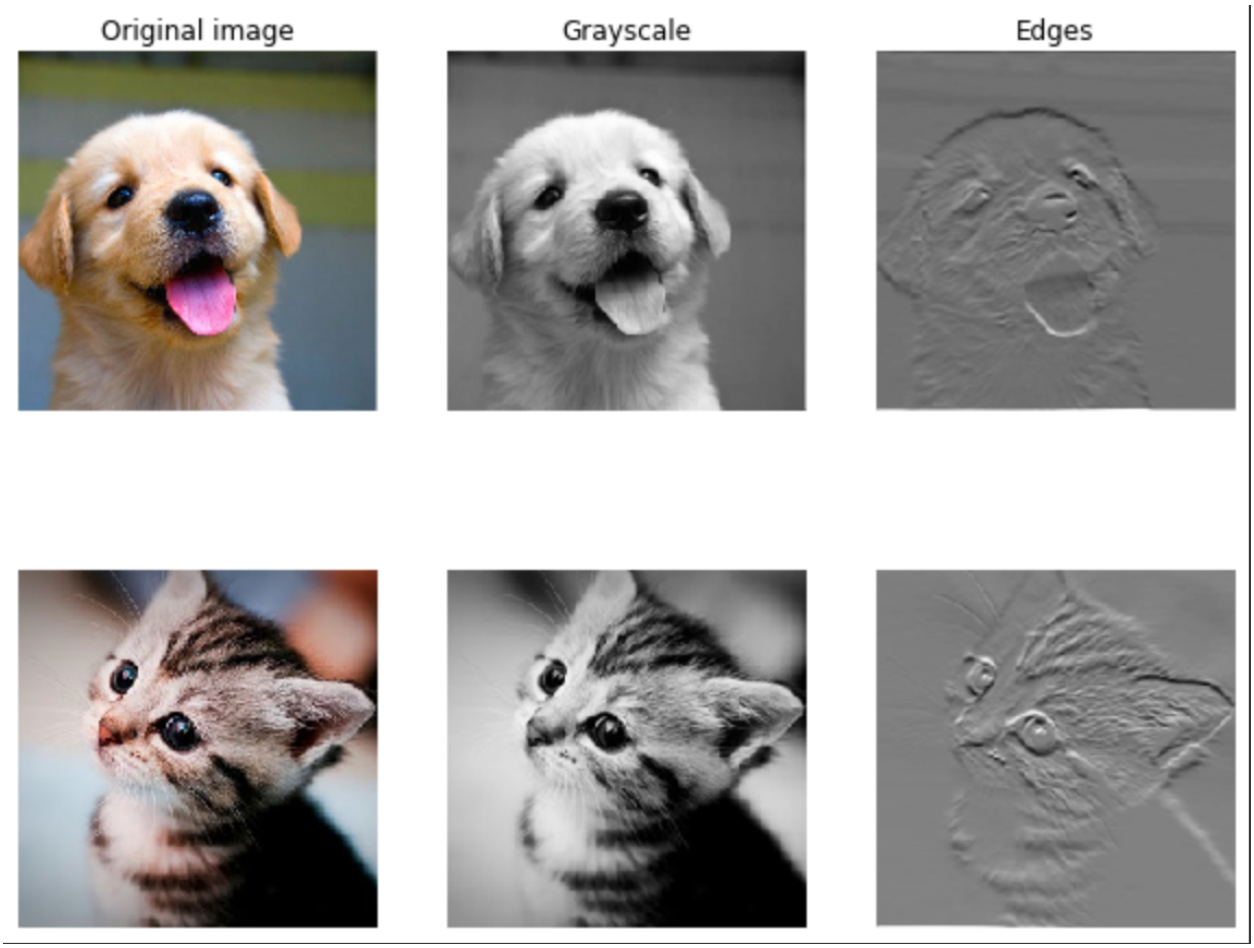

Figure 1: Results of Moving Different Kernel Over Image

4. Implement conv_backward_naive function in cs231n/layers.py . Test your implementation on the notebook. Keep fighting.

1. Input, weight, bias and convolutional parameters are obtained from cache
2. Stride and padding values are extracted from input and weight.
3. All shapes from input and weight are extracted to use them at calculating output sizes.
4. Output sizes are controlled since they have to be integers.
5. Zero padding is done to surrounding of input.
6. Output gradient arrays are generated as zeros with respect to shape of weight, bias input and padded input.
7. Finally kernel is moved around image with for loops to obtain final gradients.

5. Implement max_pool_forward_naive function in cs231n/layers.py . Test your implementation on the notebook. What are you waiting for?

1. Stride is extracted from input.
2. All shapes from input and pooling layer are extracted to use them at calculating output sizes.
3. Output sizes are controlled since they have to be integers.

4. Output array is generated as zeros with respect to output shapes.
5. Finally max pooling kernel is moved around image with for loops to obtain max pooled output.

6. Implement max_pool_backward_naive function in cs231n/layers.py . Test your implementation on the notebook. Just let it go!

1. Input x and pool parameters are obtained from the cache.
2. Stride pool height, pool width are obtained from pool parameters.
3. Output sizes are calculated with respect to pool height, pool width and stride.
4. Output sizes are controlled since they have to be integers.
5. Output gradient array is generated from calculated sizes.
6. Gradients are calculated by iterating all values in for loops.

7. There are fast implementations provided for you. Just run the cells and you will use those. They will provide a lot of speed-up. Make sure you have cython in your environment and run python setup.py build_ext --inplace . Precious!

Downloaded required packages to colab. Then, run functions for naive and fast approach.The time difference between naive and fast approach is pretty high which will save a lot of time. Also, difference between gradients are very low. For this reason it is much better to use fast approach while training the CNN.

8. Just run convolutional sandwich layers that combine multiple operations into commonly used patterns. Take my hand.

Sandwich layer which combines multiple operations are applied. Gradients are obtained from these functions. Then, their relative errors are printed as an output.

9. Implement ThreeLayerConvNet class in cs231n/classifiers/cnn.py . Run gradient check cells. Don't you let go.

9.1 Initialization

1. Depth, input_height, input_width obtained from input_dim parameter
2. First weights are generated with respect to input size and filter size. Also, weight_scale is used to keep randomly generated values in selected scale.
3. Similarly, biases are generated with respect to num_filters parameter which is given as an input from the user.
4. After first pass the layers are passed into max pool layer. Thus, max values are obtained with selected kernel and total size of input is decreased.
5. Finally, output weights are generated with respect to hidden_num, num_classes and weight scale parameter.

9.2 Loss function

1. Output of pooling layer and cache is obtained thanks to conv_relu_pool_forward function.
2. The output of pooling layer is sent to affine ReLU forward pass then output of it and cache is also calculated.
3. Lastly scores and output cache of softmax forward pass is calculated.
4. If ground truths are given from the user it start to calculate losses and gradients.
5. Firstly, predictions are passed to softmax_loss to calculate loss of predictions and softmax gradients are calculated.
6. Then, regularization losses from weights are calculated.
7. Lastly with the backpropogation gradients are obtained for each layer from output to input.

10. Overfit small data, see it on the training-validation plot. DON'T LET GO!

Model is trained on small data and overfit is achieved, since the training accuracy is 0.99 and validation accuracy is 0.22. The output of training is shown at Figure 2.

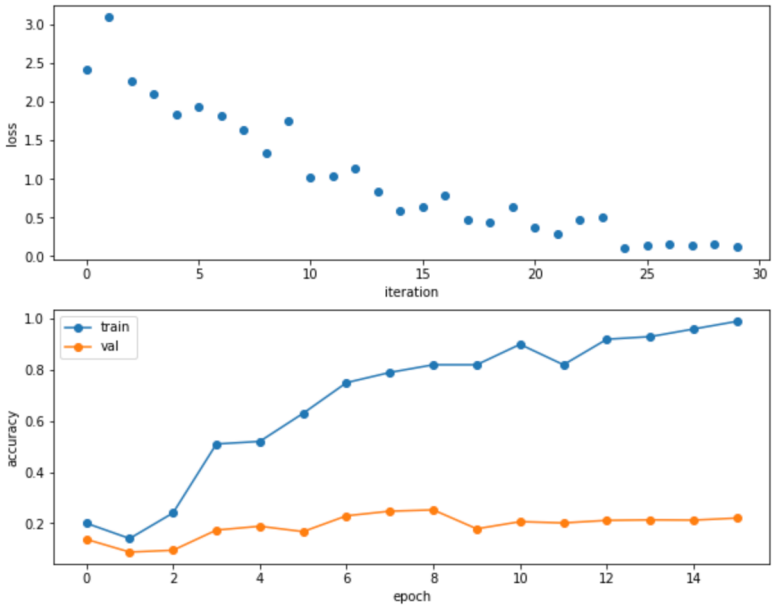

Figure 2: Training Results of Overfitted Model

11. Now train your convolutional network and acquire an accuracy over 40%. REACH!

Model is trained and train accuracy is 0.492 and validation accuracy is 0.469 . It shows that our model is not overfitted on train data. Filters are visualized at Figure 3.

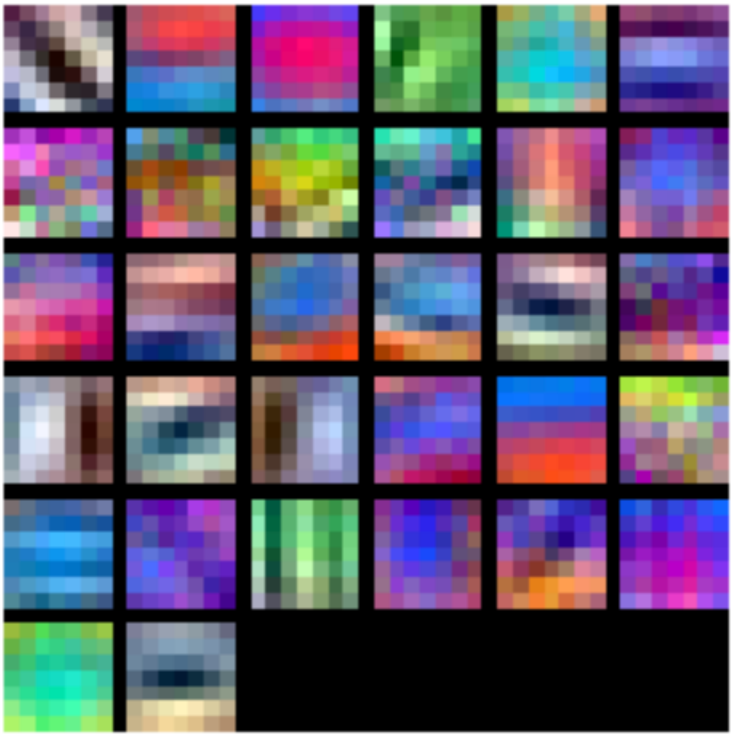

Figure 3: Visaulization of Filters

12. Now spatial normalization.. Wait! Hold on a second! What is happening?! It is.. collapsing. They retreat! YOU HAVE DONE IT! YOU ARE VICTORIOUS!

Spatial batch normalization functions are not implemented.

13. You have come to an end. We were surrounded by evil and darkness. The hope had abandoned us. We were desperate. But it is done. IT IS DONE! The rest is just being happy and enjoying for all the challenges you have overcome. Now I see the joy in your eyes. I appreciate your dauntless tenacity. And I am glad you are here with me. Here at the end of all things.

Finally, it is the end. ☺