Programming Assignment 2: Convolutional Neural Networks

Deadline: Feb. 28, 2019 at 11:59pm
Based on an assignment by Lisa Zhang

Submission: You must submit two files through MarkUs[1]: a PDF file containing your writeup, titled `a2-writeup.pdf`, and your code file `colourization.ipynb`. Your writeup must be typeset.

The programming assignments are individual work. See the Course Information handout[2] for detailed policies.

You should attempt all questions for this assignment. Most of them can be answered at least partially even if you were unable to finish earlier questions. If you were unable to run the experiments, please discuss what outcomes you might hypothetically expect from the experiments. If you think your computational results are incorrect, please say so; that may help you get partial credit.

Convolutional Neural Network for Image Processing

In this assignment, we will train a convolutional neural network to solve two classic image processing tasks: image colourization and super-resolution. First, we will focus on image colourization. That is, given a greyscale image, we wish to predict the colour at each pixel. Image colourization is a difficult problem for many reasons, one of which being that it is ill-posed: for a single greyscale image, there can be multiple, equally valid colourings.

Setting Up

We recommend that you use **Colab**(`https://colab.research.google.com/`) for the assignment, as all the assignment notebooks have been tested on Colab. Otherwise, if you are working on your own environment, you will need to install Python 2, PyTorch (`https://pytorch.org`), iPython Notebooks, SciPy, NumPy and scikit-learn. Check out the websites of the course and relevant packages for more details.

From the assignment zip file, you will find two python notebook files: `colour_regression.ipynb`, `colourization.ipynb`. To setup the Colab environment, you will need to upload the two notebook files using the upload tab at `https://colab.research.google.com/`.

Dataset

We will use the CIFAR-10 data set, which consists of images of size 32x32 pixels. For most of the questions we will use a subset of the dataset. The data loading script is included with the notebooks, and should download automatically the first time it is loaded. If you have trouble downloading the file, you can also do so manully from:

`http://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz`

To make the problem easier, we will only use the "Horse" category from this data set. Now let's learn to colour some horses!

[1] `https://markus.teach.cs.toronto.edu/csc421-2019-01`
[2] `http://cs.toronto.edu/~rgrosse/courses/csc421_2019/syllabus.pdf`

Questions

A. Colourization as Regression (2 points)

There are many ways to frame the problem of image colourization as a machine learning problem. One naïve approach is to frame it as a regression problem, where we build a model to predict the RGB intensities at each pixel given the greyscale input. In this case, the outputs are continuous, and so squared error can be used to train the model.

In this section, you will get familar with training neural networks using cloud GPUs. Open the notebook `colour_regression.ipynb` on Colab and answer the following questions.

1. Describe the model `RegressionCNN`. How many convolution layers does it have? What are the filter sizes and number of filters at each layer? Construct a table or draw a diagram.

2. Run all the notebook cells in `colour_regression.ipynb` on Colab (No coding involved). You will train a CNN, and generate some images showing validation outputs. How many epochs are we training the CNN model in the given setting?

3. Re-train a couple of new models using a different number of training epochs. You may train each new models in a new code cell by copying and modifying the code from the last notebook cell. Comment on how the results (output images, training loss) change as we increase or decrease the number of epochs.

4. A colour space[3] [1] is a choice of mapping of colours into three-dimensional coordinates. Some colours could be close together in one colour space, but further apart in others. The RGB colour space is probably the most familiar to you, the model used in `colour_regression.py` computes squared error in RGB colour space. But, most state of the art colourization models do not use RGB colour space. How could using the RGB colour space be problematic? Your answer should relate how human perception of color is different than the squared distance. You may use the Wikipedia article on color space to help you answer the question.

5. How does framing colourization as a classification problem alleviate the above problem?

B. Colourization as Classification (2 points)

We will select a subset of 24 colours and frame colourization as a pixel-wise classification problem, where we label each pixel with one of 24 colours. The 24 colours are selected using k-means clustering[4] over colours, and selecting cluster centers. This was already done for you, and cluster centers are provided in `http://www.cs.toronto.edu/~jba/kmeans_colour_a2.tar.gz` and will be downloaded automatically by the notebook. For simplicy, we still measure distance in RGB space. This is not ideal but reduces the software dependencies for this assignment.

Open the notebook `colourization.ipynb` on Colab and answer the following questions.

1. Complete the model `CNN` in `colourization.ipynb`. This model should have the same layers and convolutional filters as the `RegressionCNN`, with the exception of the output layer. Continue to use PyTorch layers like `nn.ReLU`, `nn.BatchNorm2d` and `nn.MaxPool2d`, however we will not use `nn.Conv2d`. We will use our own convolution layer `MyConv2d` included in the file to better understand its internals.

[3] `https://en.wikipedia.org/wiki/Color_space`

[4] `https://en.wikipedia.org/wiki/K-means_clustering`

2. Run `main training loop of CNN` in `colourization.ipynb` on Colab. This will train a CNN for a few epochs using the cross-entropy objective. It will generate some images showing the trained result at the end. How do the results compare to the previous regression model?

C. Skip Connections (3 points)

A skip connection in a neural network is a connection which skips one or more layer and connects to a later layer. We will introduce skip connections.

1. Add a skip connection from the first layer to the last, second layer to the second last, etc. That is, the final convolution should have both the output of the previous layer and the initial greyscale input as input. This type of skip-connection is introduced by [3], and is called a "UNet". Following the `CNN` class that you have completed, complete the `__init__` and `forward` methods of the `UNet` class.

   Hint: You will need to use the function `torch.cat`.

2. Train the "UNet" model for the same amount of epochs as the previous CNN and plot the training curve using a batch size of 100. How does the result compare to the previous model? Did skip connections improve the validation loss and accuracy? Did the skip connections improve the output qualitatively? How? Give at least two reasons why skip connections might improve the performance of our CNN models.

3. Re-train a few more "UNet" models using different mini batch sizes with a fixed number of epochs. Describe the effect of batch sizes on the training/validation loss, and the final image output.

D. Super-Resolution (1 point)

Many classic image processing problems are to transform the input images into an output image via a transformation pipeline, e.g. colourization, denoising, and super-resolution. These image processing tasks share many similarities, where the inputs are lower quality images and the outputs are the restored high-quality images. Instead of hand-design the transformations, one approach is to learn the transformation pipeline from a training dataset using supervised learning. Previously, you have trained conv nets for colourization. In this question, you will use the same conv net models to solve super-resolution tasks. In the super-resolution task, we aim to recover a high-resolution image from a low-resolution input.

1. Take a look at the data process function `process`. What is the resolution difference between the downsized input image and output image?

2. Bilinear interpolation[5] is one of the basic but widely used resampling techniques in image processing. Run `super-resolution` with both CNN and UNet. Are there any difference in the model outputs? Also, comment on how the neural network results (images from the third row) differ from the bilinear interpolation results (images from the fourth row). Give at least two reasons why conv nets are better than bilinear interpolation.

[5] `https://en.wikipedia.org/wiki/Bilinear_interpolation`

E. Visualizing Intermediate Activations (2 point)

We will visualize the intermediate activations for several inputs. Run the visualization block in the `colourization.ipynb` that has already been written for you. For each model, a list of images will be generated and be stored in `csc421/a2/outputs/model_name/act0/` folder in the Colab environment. You will need to use the left side panel (the "Table of conetents" panel) to find these images under the `Files` tab.

1. Visualize the activations of the CNN for a few test examples. How are the activation in the first few layers different from the later layers? You do not need to attach the output images to your writeup, only descriptions of what you see.

2. Visualize the activations of the colourization UNet for a few test examples. How do the activations differ from the CNN activations?

3. Visualize the activations of the super-resolution UNet for a few test examples. Describe how the activations differ from the colourization models?

F. Conceptional Problems (2 point)

1. We also did not tune any hyperparameters for this assignment other than the number of epochs and batch size. What are some hyperparameters that could be tuned? List five.

2. In the `RegressionCNN` model, `nn.MaxPool2d` layers are applied after `nn.ReLU` activations, comment on how the output of CNN changes if we switch the order of the max-pooling and ReLU?

3. The loss functions and the evaluation metrics in this assignment are defined at pixel-level. In general, these pixel-level measures correlate poorly with human assessment of visual quality. How can we improve the evaluation to match with human assessment better? You may find [4] useful for answering this question.

4. In `colourization.ipynb`, we have trained a few different image processing convolutional neural networks on input and output image size of 32x32. In the test time, the desired output size is often different than the one used in training. Describe how we can modify the trained models in this assignment to colourize test images that are larger than 32x32.

What To Submit

For reference, here is everything you need to hand in:

- A PDF file `a2-writeup.pdf` containing the following:
  - Five questions from Part A
  - Question 2 from Part B
  - Question 2,3 from Part C
  - Both questions from Part D
  - Answers to all three questions from Part E
  - Answers to all conceptual questions

  - Optional: Which part of this assignment did you find the most valuable? The most difficult and/or frustrating?

- Your implementation of `colourization.ipynb`.

This assignment is graded out of 12 points: 2 for Part A, 2 for Part B, 3 for Part C, 3 for Part D and E, and 2 for Part F.

References

[1] `https://en.wikipedia.org/wiki/Color_space`

[2] Zhang, R., Isola, P., and Efros, A. A. (2016, October). Colorful image colorization. In European Conference on Computer Vision (pp. 649-666). Springer International Publishing.

[3] Ronneberger, O., Fischer, P., and Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical Image Computing and Computer-Assisted Intervention (pp. 234-241). Springer, Cham.

[4] Johnson, Justin, Alexandre Alahi, and Li Fei-Fei. "Perceptual losses for real-time style transfer and super-resolution." In European conference on computer vision, pp. 694-711. Springer, Cham, 2016.