Assignment 4

CS5370: Deep Learning for Vision
IIT-Hyderabad
Jul-Nov 2019

Max Marks: 50
Due: 23rd Nov 2019 11:59 pm

Instructions

- Please use Google Classroom to upload your submission by the deadline mentioned above. Your submission should comprise of a single ZIP file, named `<Your_Roll_No>_Assign4`, with all your solutions, including code.
- For late submissions, 10% is deducted for each day (including weekend) late after an assignment is due. Note that each student begins the course with 7 grace days for late submission of assignments. Late submissions will automatically use your grace days balance, if you have any left. You can see your balance on the CS5370 Marks and Grace Days document.
- You have to use PYTHON for the programming questions.
- Please read the department plagiarism policy. Do not engage in any form of cheating - strict penalties will be imposed for both givers and takers. Please talk to instructor or TA if you have concerns.

1 Convolutional Neural Networks *(27 marks)*

In this problem, we will train a convolutional neural network for a task known as image colourization. That is, given a greyscale image, we wish to predict the colour at each pixel. This 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.

We recommend you to use Colab (https://colab.research.google.com/) for this assignment. 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/.

We will use the CIFAR-10 data set, which consists of images of size 32×32 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 manually from the provided `cifar-10-python.tar.gz`. To make the problem easier, we will only use the "Horse" category from this data set.

1. **Colourization as Regression *(5 marks)*:** Image colourization can be posed 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 mean-squared error can be used to train the

model. A set of weights for such a model is included with the assignment. In this question, you will get familar with training neural networks using cloud GPUs. Read the code in `colour_regression.py`, and answer the following questions.

(a) 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.

(b) 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?

(c) 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.

(d) A colour space[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 RGBcolour space is probably the most familiar to you, but most state of the art colourization models do not use RGB colour space. The model used in `colour_regression.ipynb` computes squared error in RGB colour space. How could using the RGB colour space be problematic?

(e) Most state-of-the-art colourization models frame colourization as a classification problem instead of a regression problem. Why? (Hint: what does minimizing squared error encourage?)

2. **Colourization as Classification *(2+2=4 marks)*:** 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 over colours, and selecting cluster centers. This has already been done for you, and cluster centers are provided in `colour/colour_kmeans*.npy` files. For simplicity, we still measure distance in RGB space. This is not ideal but reduces the dependencies for this assignment. Open the notebook `colourization.ipynb` and answer the following questions.

   (a) Complete the model `CNN` on `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.

   (b) 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?

3. **Skip Connections *(2+3+1=6 marks)*:** 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.

   (a) 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 results in a "UNet" model[2]. 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`.)

   (b) 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?

[1] https://en.wikipedia.org/wiki/colour_space

[2] Ronneberger et al, U-net: Convolutional networks for biomedical image segmentation, MICCAI 2015

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.

(c) 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.

4. **Super-resolution *(1+3=4 marks)*:** 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-designing 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.

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

   (b) Bilinear interpolation[3] 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. **Visualizing Intermediate Activations *(3 marks)*:** 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 `cs6360/a2/outputs/model_name/act0/` folder in the Colab environment. You will need to use the left side panel (the "Table of contents" panel) to find these images under the Files tab.

   (a) 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.

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

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

6. **Some Conceptual Questions *(2+1+1+1=5 marks)*:**

   (a) We 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. Try any one and report what you observe for the colourization problem.

   (b) 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.

   (c) 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? (*Hint:* You may find this paper useful for answering this question.)

[3] https://en.wikipedia.org/wiki/Bilinear_interpolation

(d) In `colourization.ipynb`, we trained a few different image processing convolutional neural networks on input and output image size of 32×32. 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 32×32.

2 Recurrent Neural Networks *(13 marks)*

In this problem, you will work on extending `min-char-rnn.py`, the vanilla RNN language model written by Andrej Karpathy[4]. You will experiment with the Shakespeare dataset, provided with this assignment.

1. *(2+2=4 marks)* The RNN language model uses a softmax activation function for its output distribution at each time step. It's possible to modify the distribution by multiplying the logits by a constant α:

   $$\mathbf{y} = \text{softmax}(\alpha \mathbf{z})$$

   Here, $1/\alpha$ can be thought of as a temperature, i.e. lower values of α correspond to a hotter distribution. (This terminology comes from an algorithm called simulated annealing.) Write a function to sample text from the model using different temperatures (i.e., $1/\alpha$ values). Try different temperatures, and, in your report, include examples of texts generated using different temperatures. Briefly discuss what difference the temperature makes. Include the source code of the function you wrote/modified to accomplish the task in the report. You should either train the RNN yourself, or use the weights from Part 3 (later here) - up to you.

2. *(2+2=4 marks)* Write a function that uses an RNN to complete a string. That is, the RNN should generate text that is a plausible continuation of a given starter string. In order to do that, you will need to compute the hidden activity $\mathbf{h}$ at the end of the starter string, and then to start generating new text. Include 5 interesting examples of outputs that your network generated using a starter string. (This part need not be easily reproducible). Include the source code of the function you wrote in the report. You should either train the RNN yourself, or use the weights from Part 3 - up to you.

3. *(3 marks)* The weights for a trained RNN are included as `char-rnn-snapshot.npz`. Some samples from the RNN (at temperature $1/\alpha = 1$) are included as `samples.txt`, and code to read in the weights is included as `read_in_npz.py` (if this doesn't work, try the pickle file, and get it using `import cPickle as pickle; a = pickle.load(open("char-rnn-snapshot.pkl"))`.)

   In the samples that the RNN generated, it seems that a newline or a space usually follows the colon (i.e., ":") character. In the weight data provided, identify the specific weights that are responsible for this behavior by the RNN. In your report, specify the coordinates and values of the weights you identified, and explain how those weights make the RNN generate newlines and spaces after colons.

4. *(2 marks)* Identify another interesting behaviour of the RNN, identify the weights that are responsible for it. Specify the coordinates and the values of the weights, and explain how those weights lead to the behavior that you identified.

3 Generative Adversarial Networks (GANs) *(10 marks)*

This question is adapted from the online course, CS231N, with due credit to the creators there. In this problem, you will work on learning how to implement GANs and using them for supervised classification.

[4] https://gist.github.com/karpathy/d4dee566867f8291f086

You are provided with an iPython notebook file along with MNIST dataset. Follow the instructions provided in the iPython notebook for the rest of the assignment, fill the code as required, and answer the inline questions at the end. (In case you have any issues with the datasets provided, let us know on Classroom.) *(7 marks for the code, and 3 marks for the inline questions)*