CSC 476/676 Final Project

Spring, 2021

Professor Bei Xiao

Due: April 29th

Final Presentation: April 30th

Proposal due: April 12th.

**Overview**

Final Project is an opportunity for you to apply what you have learned in class to a problem of your interest in computer vision. We recommend you work in teams. Each team can be up to 3 people.

There are two options for the final, Option 1 is doing one of the projects I recommended below. Option 2 is choosing your own project.

Please talk to me if you feel like discussing the projects. I will meet with students in the following couple of office hours.

**GPU resources (not required if you don’t do a DNN project):**

You can use free GPU with some conditions on Google Colab.

In Colab, you will get 12 hours of execution time but the session will be disconnected if you are idle for more than 60 minutes. It means that for every 12 hours Disk, RAM, CPU Cache and the Data that is on our allocated virtual machine will get erased.

So be mindful of this when you choose your training dataset and project.

Here are some tutorials:

https://medium.com/deep-learning-turkey/google-colab-free-gpu-tutorial-e113627b9f5d

https://colab.research.google.com/notebooks/gpu.ipynb

Using pytorch and cuda:

torch.cuda.device('cuda')

https://colab.research.google.com/github/d2l-ai/d2l-pytorch-colab/blob/master/chapter\_deep-learning-computation/use-gpu.ipynb#scrollTo=HnIBBFtmjWrq

**Option 1: Choose from selected Topics**

1. Image Style Transfer

A picture containing text, several

Description automatically generated

The “state-of-the-art,” while very much up to one’s artistic license, uses moment matching in deep convolutional neural networks [4]. There are several other approaches, though:

1. patch-based style transfer [3]: the gist of this method is to transfer style and color by finding and applying a patch in the source image that matches a patch in the target image. The trick is to use local image features to determine the best scale of a patch (e.g., image gradients). The proposed algorithm produces decent results but one might notice that it does not transfer edge styles and misses out on the “semantics” of a style. Perhaps you could try – using superpixel-based patches instead of squares (and making this not slow), – performing patch transfer at multiple scales (think pyramids) – finding a patch similarity metric that corresponds well to human perception
2. texture synthesis [1]: a significantly beefed (or vegetabled, if that’s more your style) up version of the previous patch-based algorithm. The results are more varied than before and may be considered to be more “visually appealing.”
3. If you read the neural style transfer paper, you’ll notice that the algorithm does not preserve color! The author has a follow-up work on augmenting the algorithm to perform color mapping color mapping [5]. The results are promising but maybe you can improve on them by applying approaches in [2] or devising one of your own!

References

[1] Michael Elad and Peyman Milanfar. “Style-Transfer via Texture-Synthesis”. In: CoRR abs/1609.03057 (2016). url: <http://arxiv.org/abs/1609.03057>.

[2] Hasan Sheikh Faridul et al. “Colour Mapping: A Review of Recent Methods, Extensions and Applications”. In: Comput. Graph. Forum 35.1 (2016), pp. 59–88. doi: 10.1111/cgf.12671. url: http://dx.doi.org/10.1111/cgf.12671. 1

[3] Oriel Frigo et al. “Split and Match: Example-Based Adaptive Patch Sampling for Unsupervised Style Transfer”. In: Proc. CVPR 2016. 2016. url: http://www.cvfoundation.org/openaccess/content\_cvpr\_2016/papers/Frigo\_Split\_and\_ Match\_CVPR\_2016\_paper.pdf.

[4] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. “A Neural Algorithm of Artistic Style”. In: CoRR abs/1508.06576 (2015). url: http://arxiv.org/abs/ 1508.06576.

[5] Leon A. Gatys et al. “Preserving Color in Neural Artistic Style Transfer”. In: CoRR abs/1606.05897 (2016). url: http://arxiv.org/abs/1606.05897.

1. Image-based material editing

Can you create an image-based method to edit “material appearance” in an image? Can you make an object appear more glossy or more translucent by adjusting statistics? We discussed one particular project, “Bandsift Material Editing[1]” and but there are other methods one cay try such as manipulating image statistics in sub-bands and color channels.

A picture containing text

Description automatically generated

There are also learning-based material editing using neural network to infer intrinsic properties’ of materials and uses them to do editing [2].

[1] I. Boyadzhiev, K. Bala, S. Paris, and E. Adelson. Bandsifting decomposition for image-based material editing. ACM TOG, 34(5):163:1–163:16,

[2] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum. Deep convolutional inverse graphics network. In NIPS, pages 2539–2547. 2015.

1. Motion Magnification of the Planets.

The planets are often called "wandering stars" because they move relative to the background stars. In fact, “wandering stars” translates to “planētes asters” in ancient Greek, from which the word “planet” is derived.

This relative motion is usually measured over the course of weeks or months. Using motion magnification, however, you should be able to show that the planets move differently than the stars even over the course of 1/2 hour or 1 hour within one night.

Take a sequence of photographs over time of the night sky with stars and one or more planets in view. A reasonable dataset will contain 200 photographs, taken 10 seconds apart. Ideally, use a camera on a tripod with a star-tracking mount. Finally, process the resulting images in a way that will exaggerate the motion of the planet(s) relative to the stars within the field of view. It would also be interesting to compare this to existing photo datasets taken over weeks or months.

Without careful research, I think this dataset might be useful:

<https://sos.noaa.gov/datasets/the-wanderers/>

There is a movie:

<https://science.gsfc.nasa.gov/690/Wanderers.html>

I might also have other astrophysical datasets if you are interested in working on this.

1. Image Inpainting

A picture containing text, different, several

Description automatically generated

**Image Inpainting** is a task of reconstructing missing regions in an image. It is an important problem in computer vision and an essential functionality in many imaging and graphics applications, e.g. object removal, image restoration, manipulation, re-targeting, compositing, and image-based rendering.

**To fill in an image with some missing parts, the simplest way is to copy-and-paste**. The core idea is to **first search** for the most similar image patches from the remaining pixels of the image itself or a large dataset with millions of images, **then directly paste** the patches on the missing parts. **However, the search algorithm could be time-consuming and it involves hand-crafted distance measure metrics**. Its generalization and efficiency still have plenty of room for improvement.

Thanks to deep learning-based approaches and the era of Big Data, we can now have data-driven deep learning-based image inpainting approaches that can generate the missing pixels in an image with good global consistency and local fine textures. This is often achieved with a conditional Generative Adversarial Networks (cGAN) by synthesizing images that reduces loss functions specified in the GAN.

Here are some popular methods:

[Globally and Locally Consistent Image Completion](https://towardsdatascience.com/a-milestone-in-deep-image-inpainting-review-globally-and-locally-consistent-image-completion-505413c300df) (GLCIC, 2017) [1] is a milestone in deep image inpainting as it defines the **Fully Convolution Network with Dilated Convolutions** for deep image inpainting and actually this is a typical network structure for deep image inpainting. By using Dilated convolutions, the network is able to understand the context of an image without employing expensive fully connected layers and hence it can handle images of different sizes.

[Patch-based Image Inpainting with GANs](https://towardsdatascience.com/revision-for-deep-image-inpainting-and-review-patch-based-image-inpainting-with-generative-4197d29c5468) [2] can be regarded as a variant of GLCIC [1]. Simply speaking, two advanced concepts namely **residual learning** [3] and **PatchGAN** [4] were embedded in GLCIC to further boost its inpainting performance. **The authors of this paper combined residual connection and dilated convolution to form a dilated residual block.** The traditional GAN discriminator was also replaced by the PatchGAN discriminator to encourage better local texture details and global structure consistency.

The core difference between traditional GAN discriminator and PatchGAN discriminator is that traditional GAN discriminator only gives a single predicted label (from 0 to 1) to indicate the realness of the input while PatchGAN discriminator gives a matrix of predicted labels (also from 0 to 1) to indicate the realness of each local region of the input. Note that each element of the matrix represents a local region of the input.

This is also an active topic of research in my own lab. So if you are interested in this topic, let me know.

Some dataset to train:

<https://github.com/NVlabs/ffhq-dataset>

<https://www.cityscapes-dataset.com/>

[1]Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa, “[Globally and Locally Consistent Image Completion](http://iizuka.cs.tsukuba.ac.jp/projects/completion/data/completion_sig2017.pdf),” *ACM Trans. on Graphics*, Vol. 36, №4, Article 107, Publication date: July 2017.

[2] Ugur Demir, and Gozde Unal, “[Patch-Based Image Inpainting with Generative Adversarial Networks](https://arxiv.org/pdf/1803.07422.pdf),” <https://arxiv.org/pdf/1803.07422.pdf>.=

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, “[Deep Residual Learning for Image Recognition](https://arxiv.org/abs/1512.03385),” *Proc. Computer Vision and Pattern Recognition* (*CVPR*), 27–30 Jun. 2016.

[4] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros, “[Image-to-Image Translation with Conditional Adversarial Networks](https://arxiv.org/pdf/1611.07004.pdf%EF%BC%89),” *Proc. Computer Vision and Pattern Recognition* (*CVPR*), 21–26 Jul. 2017.

**Option 2: Your own project.**

You could select a topic in computer vision that interests you most and work on it as your course project. Potential projects could be based on applications and models:

* Applications: You would apply the techniques of computer vision to some specific applications with your background and interest, such as some image processing app and video recognition software. For example, you can apply several CNN architectures (AlexNet, ResNET50, VGG19, Simple Frame work of Contrastive Learner) for classification task.
* Models: You would build up some new models, or improve previous models or methods, then evaluate the proposed models systematically on some standard image datasets to show the improvement. This is challenging. But you can think of very incremental improvements over existing methods. Usually, you can start by implementing an existing method in a paper and try it out on a new task.

Please clearly specify and justify: (1)what will be the approach; (2)why is it interesting; (3) how will you evaluate success. Before proceeding this option, please find teammates through Piazza then draft a. project proposal together and send it to the instructors for plausibility analysis, then set up a meeting about the project detail.

**Summary of the project proposal.**

The project proposal should be one page maximum following this template:

• What is the problem/question that you will be investigating?

• What are the most relevant readings? (2-4 papers)

• What data will you use?

• What are the existing methods? Are their implementations available?

• What method or algorithm will you use?

• How will you evaluate your results?

• Qualitatively, what kind of results do you expect (e.g. plots or figures)

• Quantitatively, what kind of analysis will you use to evaluate and/or compare your results (e.g. what performance metrics or statistical tests)?

## **Final Report**

The report should be 3-4 pages CSC 476 (Undergrads), and 5-6 pages for CSC676 (the upper limit of 6 pages is strict!), including references in [CVPR format](http://www.pamitc.org/cvpr13/files/cvpr2013AuthorKit.zip). It should be structured like a research paper, with sections for introduction, related work, the approach/algorithm, experimental results, conclusions and references. Project reports should be individually submitted and the contributions of each team member should be clearly described.

Regarding the reports:

* Each student should submit an individual copy. All the members of the group can share figures and text. But each copy should have one section that will be individual and should describe the contribution made by the student.
* The rest of the document can be identical across members.
* Each copy should include the names of all the collaborators.
* Use Figures and Tables to your advantage.

You should describe and evaluate what you did in your project, which may not necessarily be what you hoped to do originally. A small result described and evaluated well will earn more credit than an ambitious result where no aspect was done well. Be accurate in describing the problem you tried to solve. Explain in detail your approach, and specify any simplifications or assumptions you have taken. Also demonstrate the limitations of your approach. When doesn’t it work? Why? What steps would you have taken have you continued working on it? Make sure to add references to all related work you reviewed or used.

You are allowed to submit any supplementary material that you think it important to evaluate your work, however we do not guarantee that we will review all of that material, and you should not assume that. The report should be self-contained.

**Submission:** submit your report to stellar as a pdf file named <your\_firstname\_lastname>.pdf. Submit any supplementary material as a **single zip file** named <your\_firstname\_lastname>.zip. Add a README file describing the supplemental content.

**Your code**

You must submit original codes that generated the results in the report. All the codes must be uploaded to GITHUB. You can choose to submit a zipped folder of .py codes or Google CoLab. Each group can submit just the same code.

## **Grading Policy**

The following is the weight for two parts:

* Research component of final project (80%), this will be graded individually based on the individual’s report.
  + Abstract (5%)
  + Introduction (10%)
  + Related work (10%)
  + Approach (and technical correctness) (10%)
  + Experimental results (and technical correctness) (10%)
  + Conclusion (5%)
  + References (5%)
  + Codes, accuracy, reproducibility, and style (20%)
  + Overall clarity of the report (5%)
* Final presentation (20%), One presentation per team.

Since the grading is assigned on both team project presentation and the individual report, it is possible each student receive individualized grade.