Deep Learning for Visual Recognition
MA-INF 2313, WS 2019/20
Course Project Descriptions

The course project is a chance for you to get some hands-on experience working with a few recent deep networks. You will be asked to setup, train and test your own neural network on real world datasets. You are welcome to implement it any deep learning framework of your choice or use existing available code online. **For the projects, you will work in a team of two groups**.

1 Topics

There are three possible topics for the projects:

1. Classification: train a CNN for classifying 101 different food categories.
2. Stylization: train a CNN for artistically stylizing photos.
3. Synthesis: train a generative adversarial network for synthesizing novel images.

2 Presentation

You will present your project in a **short 10 minute presentation**. The presentations should include

- motivation and objectives of the project (restricted to 1-2 slides)
- a short overview of the theoretical basis and network architecture (restricted to 1-2 slides)
- your experiences in training the network: what worked, what didn't?
- experimental results
- other interesting insights

3 Grading

The projects will be graded 50% on implementation and 50% on the presentation. You'll be asked to hand in your source code and slides on your designated presentation date. You must receive a pass on the project (50% overall) to be permitted to take the final exam. At least one member from each group must be present on the day of the presentation.

4 Computing & GPUs

Each group will be provided an account to a GPU server for running experiments for the projects. The account credentials as well as a small tutorial will be released shortly on the course website. You can additionally explore platforms such as Google Collab, AWS, vast.ai. Few of these platforms have the option of training your model on the cloud for free.

Project 1: Classification

In this project, you will use features from off-the-shelf pre-trained CNNs to classify different types of food. You will then learn to fine-tune your own CNN for classification. To learn more about using CNN features and fine-tuning, please read the papers [1, 2],the Caffe guide [3] and the explanation on transfer learning from Stanford cs231 [4]. You will be using the Food-101 dataset [5], which has 101 food categories. Each category has 1000 images, 750 for training and 250 for testing. Training images contain some noise from intense colours and sometimes wrong labels; test images are clean and manually reviewed.[1]

Figure 1: Sample images from the Food-101 dataset.

Tasks For each of the following tasks, select randomly 3 subsets of 10 and 2 subsets of 30 classes and use the same splits for all the experiments. Report the average classification accuracy over the 3 random splits of 10 classes as well as the 2 random splits of 30 classes. For each task, plot the confusion matrix for one of the 3 random splits consisting of 10 classes and report the classification accuracy for each class individually; use the same split for plotting the confusion matrix for the tasks as well. Finally, compute the classification accuracies with training and testing on all of the 101 classes. Report the top 5 classes with highest classification accuracy and the 5 worst performing classes. Explain your results (certain classes share visual similarity making it harder for the networks to reason and distinguish etc.).

1. Using the responses from the final fully connected layer as features, train a **linear SVM** to perform classification. You can use the lib-SVM[2] package which is available for both C++ and Python. Compare the classification accuracy on the features obtained from using off-the-shelf VGG-16[3] and ResNet-34[4].

2. Starting with the base VGG-16, fine tune the network to classify the same 10 and 30 splits from the previous part. Also fine tune the network to classify all the 101 classes. Repeat the same experiment with ResNet-34.

3. In the lecture on regularization, we discussed explicit handling of incorrect ground truth labels in the training via label smoothing regularization. In this regularization scheme, the factor ϵ is an estimate of how frequently you expect the label to be incorrect. You can find out more details about this in Section 7 of the work by Szegedy *et al.* [6]. Incorporate label smoothing regularization into your fine-tuning scheme for both VGG and ResNet and apply to the split considering all 101 classes. Do the classification results get better or worse ?

[1] You can also use pre-trained VGG-16 and ResNet-34 provided by Tensorflow or any library of your choice.
[2] `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`
[3] `http://www.robots.ox.ac.uk/~vgg/research/very_deep/`
[4] `https://github.com/pytorch/vision/blob/master/torchvision/models/resnet.py`

Project 2: Stylization

In this project you are going to work on style transfer. The idea is to take an existing photo, and render it in a specific characteristic style, e.g. as per a famous painting. For example in Figure 2 the Pablo Picasso painting's style is transfered to the photo of the baby. You can see many more examples at deepart.io [5].

To perform style transfer, one needs to consider preserving the original image's content and at the same time adapt stylistic characteristics. To learn more about stylization, please read [7], [8]. You can use the implementation of Johnson *et al.* [7] on Torch[6]. You can use other available implementations on Github as well.

Figure 2: Example results for style transfer [7].

Tasks

1. Train your own models with the same styles as in [7] and try to recover similar outputs.
   (a) Test your models for different types of images e.g noisy image vs. clean (add Gaussian or salt and pepper noise to your image), with lots of clutter or objects vs. less cluttered images, and indoor vs. outdoor scenes. Are there any noticeable differences in output?
   (b) Test your models on the same images but at different resolutions. How is the output affected?
2. Train your own models with three different styles. Train one model with Femme nue assise by Pablo Picasso, one model with Composition VII by Wassily Kandinsky and a third style of your own choosing.
3. As discussed in [9], using instance normalization instead of batch normalization significantly improves the quality of feed forward style transfer models. Train your models with instance normalization and show how instance normalization changes the style transfer models qualitatively in comparison to batch normalization.
4. In the algorithm there are two losses. One is the content loss (a feature reconstruction loss) and the other is the style loss (style reconstruction loss). The stylization network is trained using stochastic gradient descent to minimize a weighted combination of these

[5] `https://deepart.io`
[6] https://github.com/jcjohnson/fast-neural-style/blob/master/doc/training.md

loss functions. For the following tasks (a) and (b) use the image of The Starry Night by Vincent van Gogh which is provided in the source code.

(a) Train a model using only the style loss. How are the outputs affected?

(b) Train models with different weightings of the content and style losses. How does these different weightings affect the stylization qualitatively? You can check the default weights in the source code and can try out different weightings.

5. Rather than encouraging the pixels of the output image to exactly match the pixels of the target image, the authors of [7] compute their loss based on the activations of some layers of the network. For all style transfer experiments they compute feature reconstruction loss at layer `relu3_3` and style reconstruction loss at layers `relu1_2`, `relu2_2`, `relu3_3`, and `relu4_3` of the VGG-16 [10] loss network. Experiment with computing this loss at different layers and report the differences in the style transfer with your insights.

6. During training, keep track of the feature reconstruction loss and style reconstruction loss over iterations and plot them. You can visually check the progress of the style transfer by performing style transfer to any test image at any iteration.

Project 3: Synthesis

In this project you are going to work on generating images by training Generative Adversarial Networks (GANs) [11]. You will use the implementation of Radford *et al.* [12] on Torch [7]

Figure 3: Generated bedrooms [12].

Tasks

1. One common technique for evaluating the quality of unsupervised representation learning algorithms is to apply them as a feature extractor on supervised datasets and evaluate the performance of linear models trained using these features. Do classification on Food-101 dataset [5] (see Project 1) using GANs trained on the IMAGENET-1K as a feature extractor[8]. Train a **linear SVM** for classification using these features and report your classification results. You can use the lib-SVM package

2. Train a generator of yourself using another dataset. You are free to choose any one of the datasets listed **here** except the Bedroom dataset since it is used in the paper.

3. Train a generator of yourself using any dataset and or topic you prefer. Generate images using your own trained GAN, and try to get similar vector arithmetics as introduced in [11] (see 'vector arithmetic in face samples' section). Feel free to create your own dataset of interest; other possibilities include Caltech-UCSD Birds-200-2011, Oxford-102 flowers and Food-101.

[7] https://github.com/soumith/dcgan.torch

[8] You can also opt for a smaller dataset such as CIFAR-10 or CIFAR-100.

References

[1] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. *CoRR*, 2014. http://arxiv.org/abs/1403.6382.

[2] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *CoRR*, 2014. http://arxiv.org/abs/1411.1792.

[3] Fine-tuning CaffeNet for Style Recognition on Flickr Style Data. http://caffe.berkeleyvision.org/gathered/examples/finetune_flickr_style.html.

[4] Transfer Learning. http://cs231n.github.io/transfer-learning/.

[5] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *ECCV*, 2014. https://www.vision.ee.ethz.ch/datasets_extra/food-101/static/bossard_eccv14_food-101.pdf.

[6] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015. http://arxiv.org/abs/1512.00567.

[7] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, 2016. http://cs.stanford.edu/people/jcjohns/papers/eccv16/JohnsonECCV16.pdf.

[8] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015. http://arxiv.org/abs/1508.06576.

[9] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016. http://arxiv.org/abs/1607.08022.

[10] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. http://arxiv.org/abs/1409.1556.

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *NIPS*, pages 2672–2680. 2014. http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

[12] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015. http://arxiv.org/abs/1511.06434.