University of Michigan
EECS 598-012: Unsupervised Computer Vision
Winter 2021. Instructor: Andrew Owens.

Problem set

Posted: March 14, 2021 **Due:** April 21, 2021

Please submit your solution to Gradescope as a .zip file, including all .ipynb (containing the visualizations that we requested) and *.py files. Your .zip file should be named as <unique name>_<umid>.zip. Example: adam_01100001.zip. Also, please remember to put your name and unique name in the first text block of the notebook.

Credits:

- Problems 1-3 are based on UC Berkeley's CS294-158 class, with instructors Pieter Abbeel, Peter Chen, Jonathan Ho, and Aravind Srinivas.
- The Colab notebook code and instructions are partially based on Justin Johnson's EECS 598-005 class.

To get started:

- Download the code `EECS598-012-ProblemSets.zip`, unzip it, and upload the folder to your Google Drive.
- Download the `tiny_voxceleb.tar.gz` file (around 7 GB), and upload it to the folder `EECS598-012-ProblemSets/dataset` in your Google Drive.
- Open the `.ipynb` notebook file in Google Colab. You can find this option by right-clicking the file from Google Drive.

Problem 1 *Autoregressive models* (10 pts)

You'll implement PixelCNN [7], a popular auto-regressive model. We recommend referring to the paper throughout the problem. The notebook `pixel_cnn.ipynb` will guide you to implementation. We've given you a partial implementation, which you'll complete as follows:

- Implement the loss function. Recall that PixelCNN treats pixel intensities as discrete categories (3 pts).
- Implement the autoregressive sampling procedure, where each pixel is predicted conditioned on the previously generated content (3 pts) .
- Implement masked convolution (2 pts).

- Visualize the average negative log-likelihood during training, using the given code (1 pt).
- Visualize the generated images (1 pt).

Problem 2 *Variational autoencoders* (20 pts)

Next, we'll implement a variational autoencoder [4], along with a vector-quantized VAE (VQ-VAE) [6]. The notebook `vae.ipynb` will guide you through implementing and training the VAE and VQ-VAE models. We recommend doing Problem 1 before implementing the VQ-VAE, since it requires a PixelCNN. Your implementation/visualization will include the following items:

VAE model:

- Derive the VAE loss functions, following the instructions we provide (Q1.1, 3 pts).
- Implement the loss function (2 pts).
- Implement the forward pass of `VAEConvNet` (1 pt).
- Plot the training losses (1 pt).
- Visualize the generated images (1 pt).
- Visualize the images together with their reconstructions (1 pt).
- Interpolate between different latent variables and use the decoder to generate the corresponding images (1 pt).

VQ-VAE model:

- Implement the VQ-VAE loss function (2 pts).
- Implement the vector quantization layer (2 pts).
- Fill in the code for the PixelCNN model. This should just involve copy-pasting code from Problem 1 (2 pts).
- Plot the training losses for the VQ-VAE (1 pt).
- Plot the training losses for training the PixelCNN model (1 pt).
- Visualize the generated images (1 pt)
- Visualize the images together with their reconstructions (1 pt).

Problem 3 *Bidirectional generative adversarial networks (BiGANs)* (10 pts)

Recall that a BiGAN is a variation of a generative adversarial model (GAN) that can both encode images and generate them [2, 3]. The notebook `bigan.ipynb` will guide you through implementing this model. You will do the following:

- Implement the loss function for training the generator, encoder, and discriminator (3 pts).
- Plot the training losses (2 pts).
- Visualize the generated images (2 pts).
- Visualize the images together with their reconstructions (1 pt).
- Plot the training losses for the linear classifiers, trained on both the learned encoder and a randomly initialized encoders (2 pts).

Problem 4 *Contrastive learning* (10 pts)

We'll implement an audio-visual contrastive learning model, and use it for sound source localization. The model is a simplified version of [1]. We have replaced the loss function with the InfoNCE loss [5]. We train the model using contrastive learning. After training, we perform sound source localization by taking the dot product between visual and audio features. Locations within an image that have a large dot product with the audio are highly informative about the sound, and are likely to correspond to sound sources.

The notebook `av_loc.ipynb` will guide you through the training of the audio-visual model. Your implementation/visualization will include the following steps:

- Implement the contrastive loss function (4 pt).
- Plot the losses during training (3 pt).
- Visualize the sound source localization results as a heat map (3 pt).

References

[1] T. Afouras, A. Owens, J. S. Chung, and A. Zisserman. Self-supervised learning of audio-visual objects from video. *European Conference on Computer Vision (ECCV)*, 2020.

[2] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

[3] J. Donahue and K. Simonyan. Large scale adversarial representation learning. *arXiv preprint arXiv:1907.02544*, 2019.

[4] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[5] A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[6] A. v. d. Oord, O. Vinyals, and K. Kavukcuoglu. Neural discrete representation learning. *arXiv preprint arXiv:1711.00937*, 2017.

[7] A. Van Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pages 1747–1756. PMLR, 2016.