CS 294 Checkpoint 1:
Variational Auto-encoders and other Generative Models

Project members: Jianbo Chen Billy Fang Cheng Ju

26 September 2016

Team members and roles

- Jianbo Chen: variational auto-encoder, overall infrastructure
- Billy Fang: conditional variational auto-encoder, semi-supervised learning model
- Cheng Ju: DRAW (generation with attention models)

Problem statement and background

Variational auto-encoders (VAEs) [4] are a class of deep generative models based on a variational method in Bayesian inference. There exist many variants including conditional VAEs [1], VAEs tailored for semi-supervised learning [3], attention-based generative models [2], and more.

One obvious feature of generative models is their ability to produce realistic images and other output. With sophisticated models, one can even replicate consistent "style." For example, DRAW [2] was able to generate images of house numbers with number combinations not seen in the training set, but with a consistent style/color/font of street sign in each image. Additionally, as models learn to generate realistic output, they learn important features along the way, which potentially can be used for classification; we consider this in the conditional VAE and semi-supervised learning models.

As we implement known models, we also explore generalizations and modifications. We have considered modifying the architecture of the various networks, modifying the various ways one can incorporate side data in the conditional VAE and semi-supervised settings, as well as considering non-isotropic Gaussian models (which the current literature eludes).

Since we are currently at an implementation/exploratory stage, we have not rigorously evaluated the quality of our generated output, but we intend to be more quantitative in this regard later on.

Data source

Currently we are using the MNIST handwriting dataset through TensorFlow. We hope to move on to natural images in the future.

Models

1. **Variational auto-encoder (VAE).** We start from the model used in [4]. We have implemented two types of variational auto-encoder:

(a) Baseline model: The first VAE uses single-layer networks for both the encoder and the decoder, which we call fc-VAE. The following is a sample of some of the generated images.

(b) Convolutional model: The second VAE uses convolutional networks for the encoder and uses deconv nets for the decoder, which we call conv-VAE. A sample of the generated images appears below. Concretely, our encoder is of the form:

conv-BN-ReLU-conv-BN-ReLU-linear-BN-ReLU.

The decoder is of the form:

Linear-ReLU-deconv-BN-ReLU-deconv-BN-RelU.

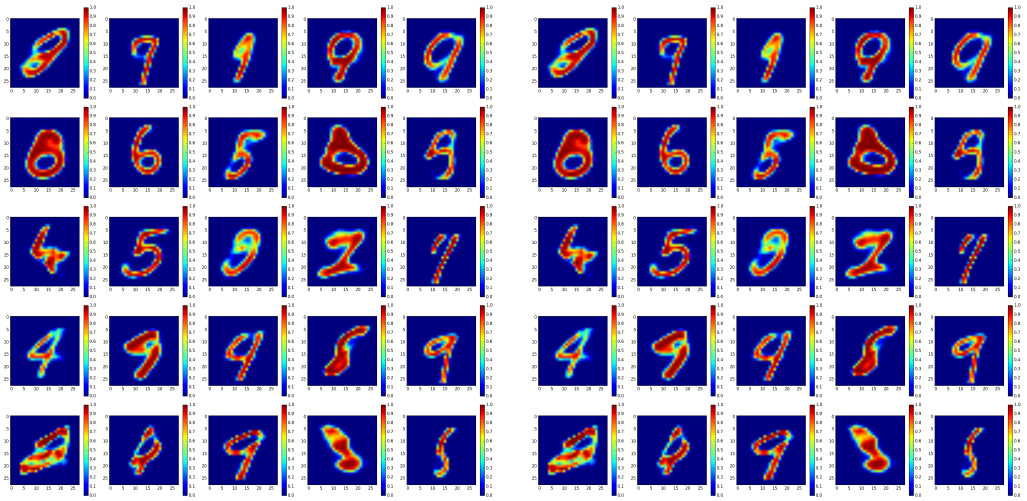

Figure 1: Output of fc-VAE (left) and conv-VAE (right)

2. **Conditional VAE (CVAE).** We followed the model described in [1], with fully-connected networks, which essentially is the same as the vanilla VAE, but also takes side information such as partial images or labels. Below we show that, given half of an image, it is able to generate an image that matches the given half. A similar model is able to take a label and generate the corresponding digit, with varying styles.

3. **DRAW.** This model is based on [2]. The baseline model is a two-LSTM network: the first LSTM is a encoder, which takes an image as input and outputs a low-dimensional vector z. The second LSTM is a decoder, with takes z as an input and outputs an image.

4. **Semi-supervised model.** This model is from [3]. It is slightly more versatile than the CVAE in that it can simultaneously learn from a dataset where not all examples are labeled. We are still setting this up.

Challenges

At first we thought replacing fully connected layers with convolutional and deconvolutional layers will improve the results. But then we found it was not the case. Currently the simplest VAE yields the best results with the chosen parameters. We are still tuning the parameters and

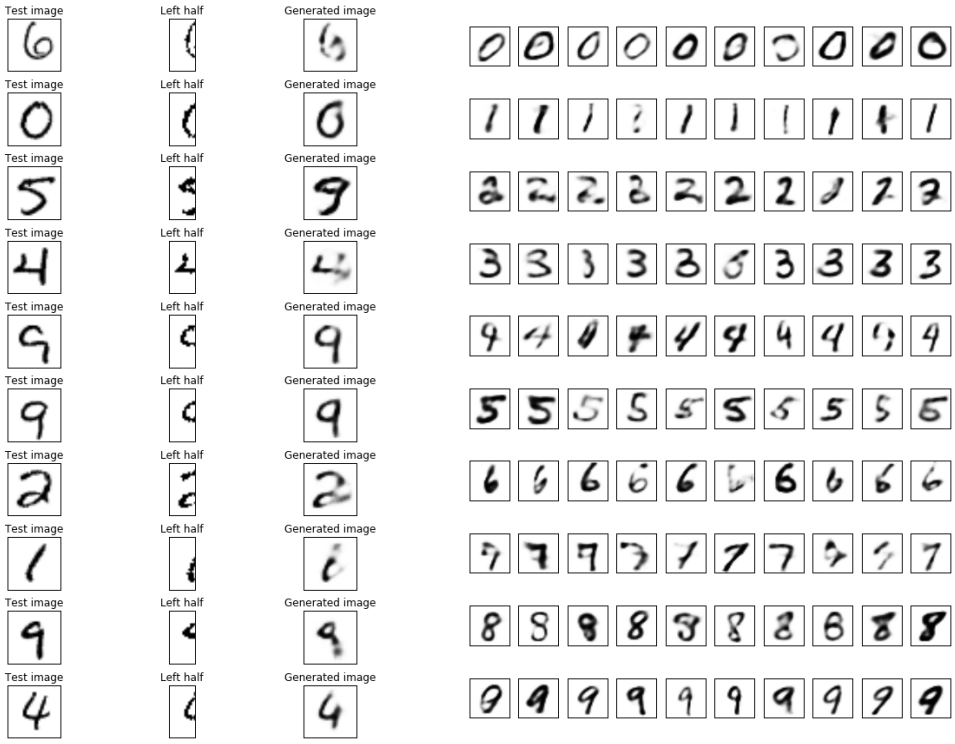

Figure 2: CVAE: Completing half of an image (left), and generating given labels (right)

try to find out whether it is the problem of conv-VAE itself or it is because we select a bad set of parameters.

Implementing the semi-supervised model of [3] was a little tricky because we need to handle labeled and unlabeled data differently.

In all models we still need to find good architectures and tune hyperparameters.

Tools and related challenges

We are currently using TensorFlow and collaborating with Github.

When we run our experiments on AWS machines, we only receive the final states of the checkpoint. This makes it hard to tune the parameters. Currently we are adding TensorFlow functions to our code that can report the loss history (in our case we report the encoder loss and decoder loss separately).

The checkpoint files are very large (100-200 MB). Therefore it is difficult to download and upload those files. Further, the naming of the checkpoint files is important. We need to make a consistent rule to name those files when doing experiment separately.

Evaluation

Since we are in the nascent stages of our project, we are currently just using eyesight to judge the quality of our generated images. In the future, we may consider other measures such as log-likelihood, or adopt an adversarial approach by applying a discriminator/classifier on our images.

References

[1] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[2] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[3] Diederik P. Kingma, Danilo Jimenez Rezende, Shakir Mohamed, and Max Welling. Semi-supervised learning with deep generative models. *CoRR*, abs/1406.5298, 2014.

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