Deep Learning

Variational Auto-Encoders

Author: Yantian Luo

Institute: Electronic Engineering

ID: 2018310742

Update: June 11, 2019

Contents

Chapter 1 Introduction

In this homework, we need to use Zhusuan to implement Variational Auto-Encoders. Let x denote the observable data and z denote the corresponding latent variable that can describe the data. The generative process of the Variational Auto-Encoders (VAEs) with Gaussian prior of z and Bernoulli likelihood of x is defined as follows:

$$\begin{aligned} z &\sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\ \mu_x &= f(z) \\ x &\sim \mathcal{B}(\mu_x) \end{aligned} \tag{1.1}$$

where $\mathcal{N}(0, I)$ is the standard Gaussian distribution, f is parameterized by a deep neural network and $\mathcal{B}(\mu_x)$ is the multivariate Bernoulli distribution. As in our case where x is multi-dimensional, each dimension of x is sampled independently given the corresponding dimension of μ_x following one-dimensional Bernoulli distribution.

Also we can define the generative process of the Variational Auto-Encoders (VAEs) with Gaussian prior of z and Gaussian likelihood of x as follows:

$$\begin{aligned} z &\sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\ \mu_x &= f_\mu(z) \\ \sigma_x &= f_\sigma(z) \\ x &\sim \mathcal{N}(\mu_x, \sigma_x) \end{aligned} \tag{1.2}$$

Then to infer the latent variable z given observable x, VAEs build a recognition model as follows:

$$\begin{aligned} \mu_z &= g_\mu(x) \\ \sigma_z &= g_\sigma(x) \\ z &\sim \mathcal{N}(\mu_z, \sigma_z) \end{aligned} \tag{1.3}$$

where g_μ and g_σ are parameterized as deepneural networks, which can share most of the parameters.

To jointly learn the parameters in all networks, VAEs defines a variational lower-bound of the marginal data likelihood for per data, i.e. $p(x)$ and sums over them.

Chapter 2 VAE-Base Results

In this chapter, we implement the code “vae_base.py” and add code to randomly sample 100 latent variables z from Gaussian prior and plot their corresponding images. We training for 10 epochs and draw the corresponding images as shown in Figure 2.1. We can find that the generating images is similar with the true images. And we also draw the corresponding images for some epochs as shown in 2.2, from the training process images, we can find that the images is getting clearer and more and more similar with true images.

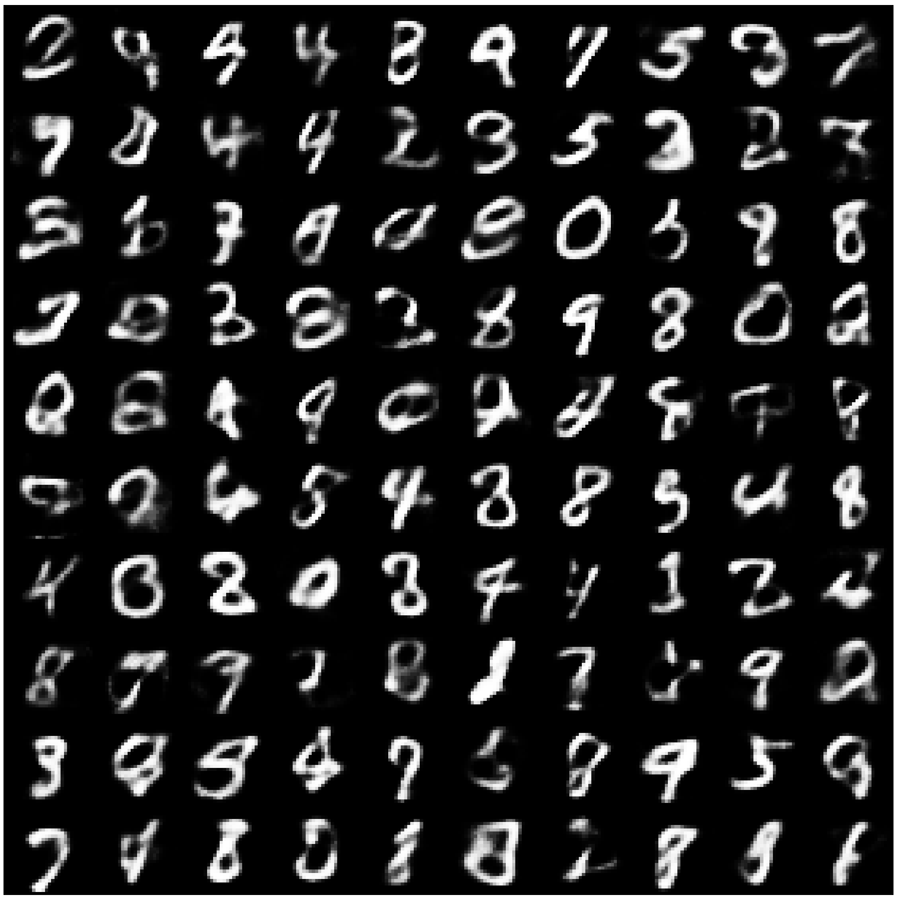
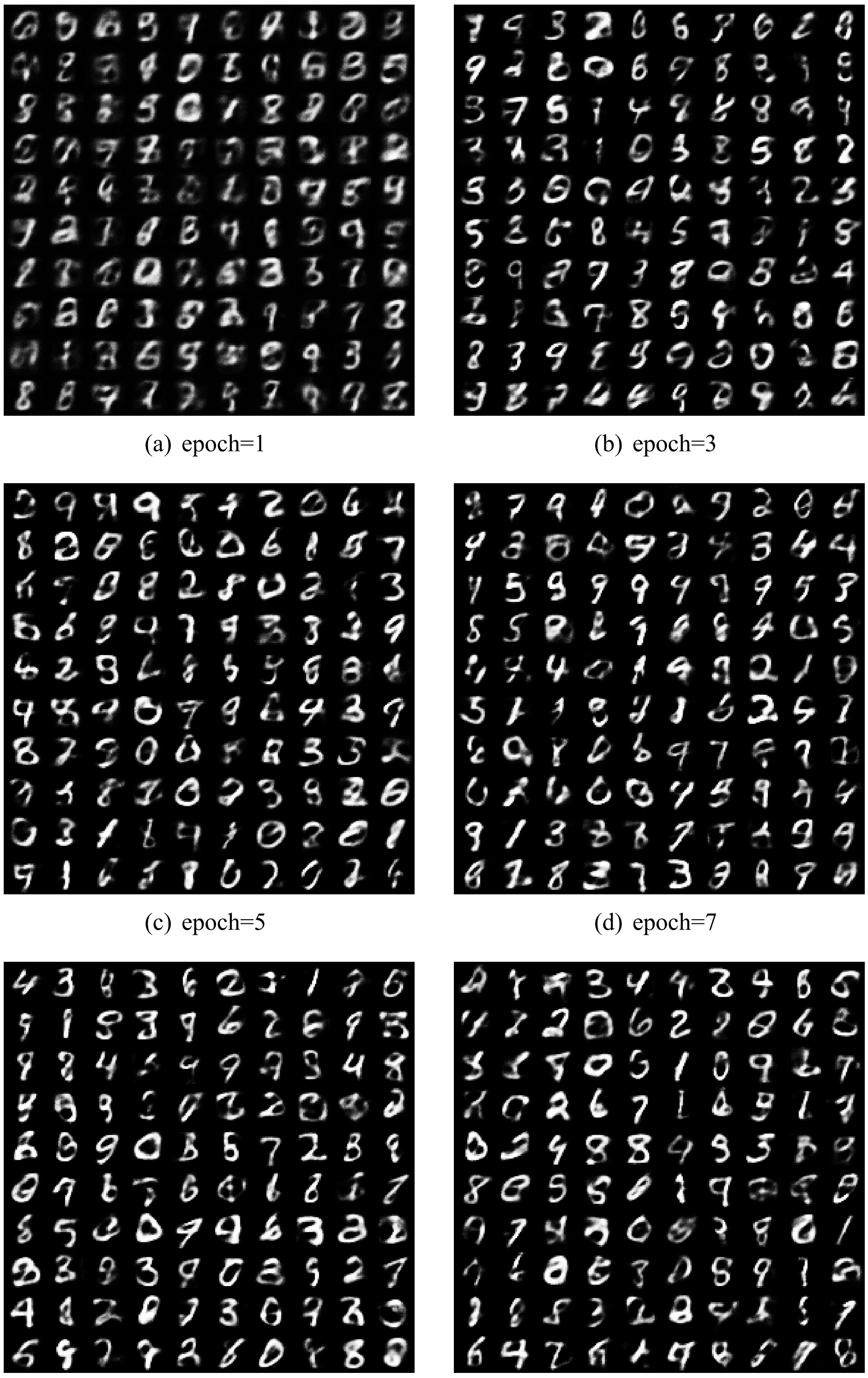


Figure 2.1: The corresponding images after training



(a) epoch=1 (b) epoch=3

(c) epoch=5 (d) epoch=7

(e) epoch=8 (f) epoch=9

Figure 2.2: The corresponding images for some epochs

Chapter 3 VAE-Gaussian Results

3.1 The Providing Codes

In this section, we implement the code “vae_gaussian.py” and add code to randomly sample 100 latent variables z from Gaussian prior and plot their corresponding images. We training for 10 epochs and draw the corresponding images as shown in Figure 3.1. We can find that the generating images much worse than Bernoulli likelihood case. The results may be that Bernoulli likelihood will bring some prior information to x but Gaussian doesn’t. And we also draw the corresponding images for some epochs as shown in 3.2, from the training process images, we can find that the images is getting clearer but also not similar with true images in 10 epochs.

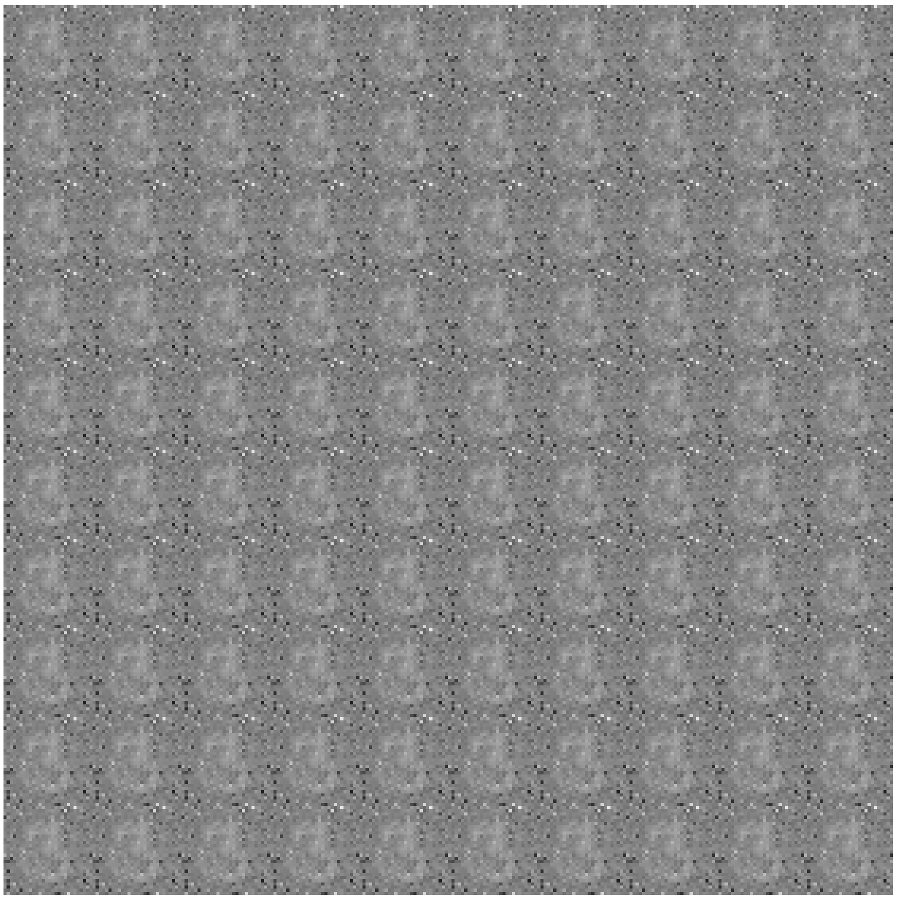
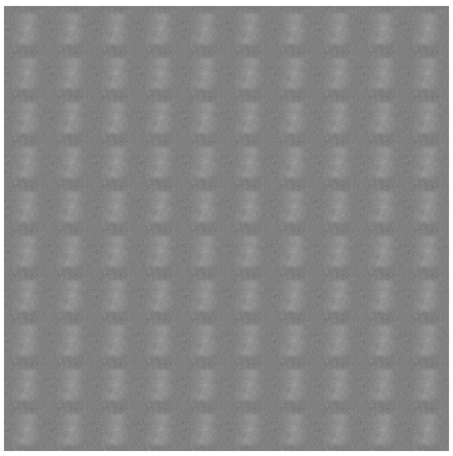
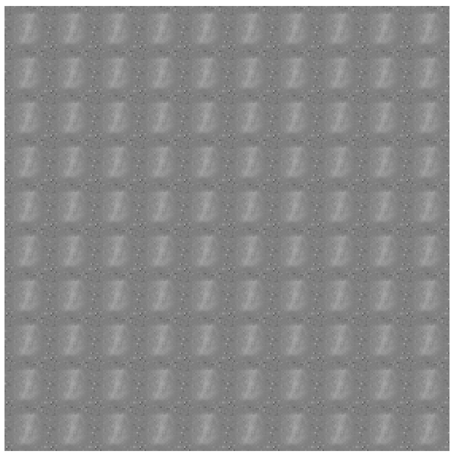
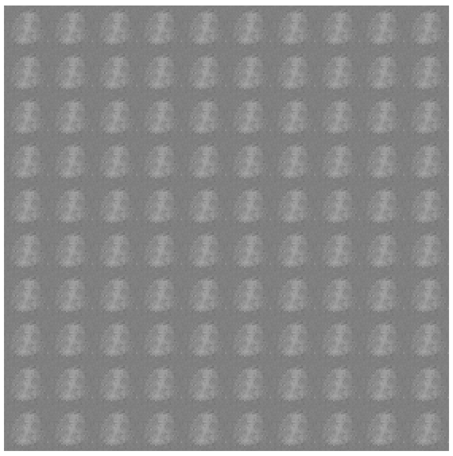
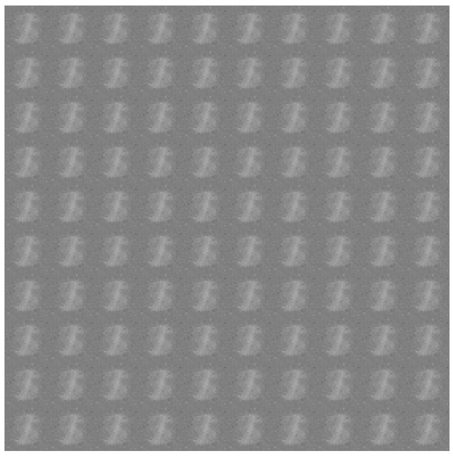


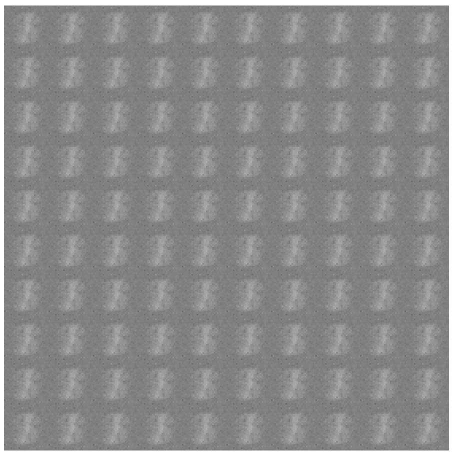
Figure 3.1: The corresponding images after training


(a) epoch=1


(b) epoch=3


(c) epoch=5


(d) epoch=7


(e) epoch=8

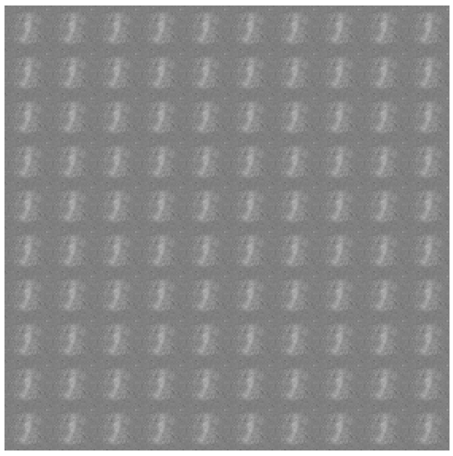

(f) epoch=9

Figure 3.2: The corresponding images for some epochs

3.2 Another implement

According the results in the last section, we can find that the results is not very good. But I don't know whether the code is right. Therefore, in this section, we use codes in "vae.py" which is similar with the tutorial codes provided by zhusuan to implement. We training for 10 epochs and draw the corresponding images as shown in Figure 3.3. We can find that in this implement, the results is better than the providing codes but also is worse than Bernoulli likelihood case. And we also draw the corresponding images for some epochs as shown in 3.2, from the training process images, we can find that the images is getting clearer in 10 epochs.

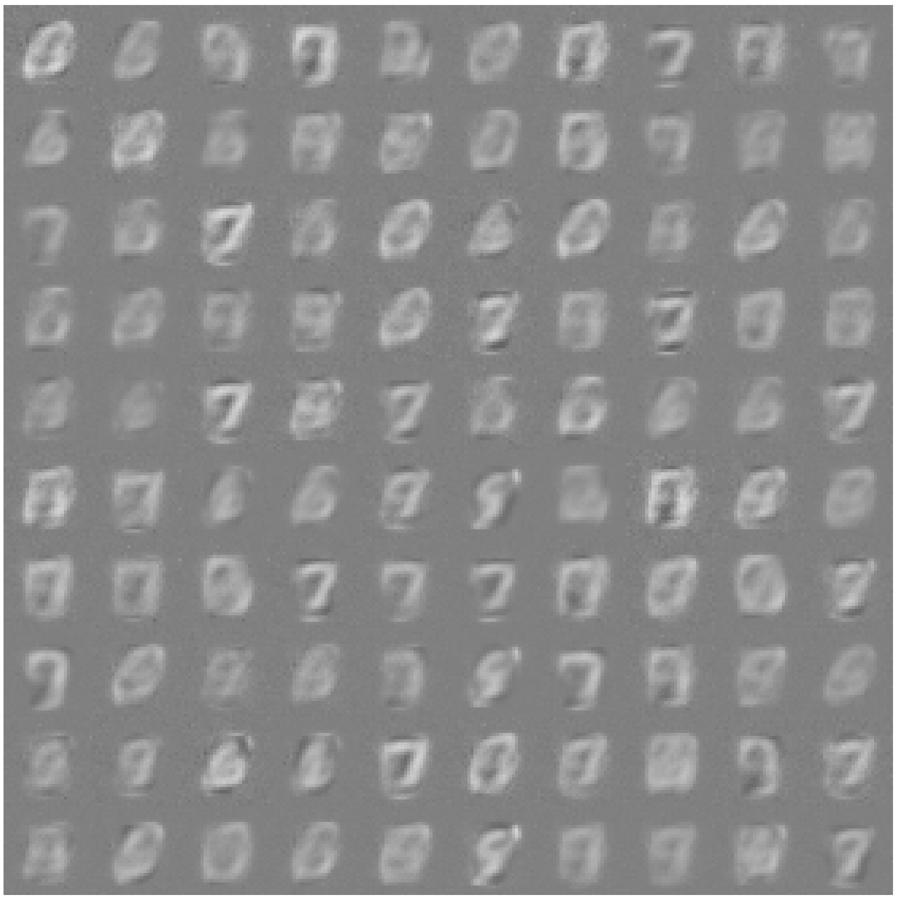
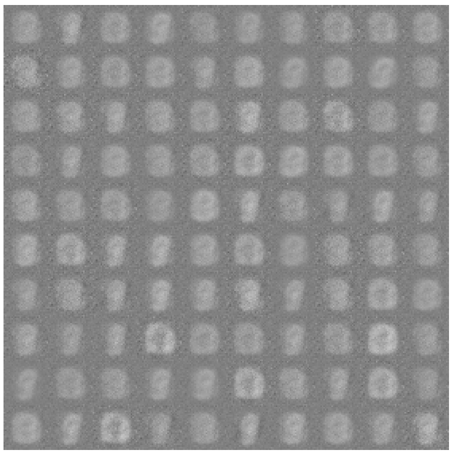
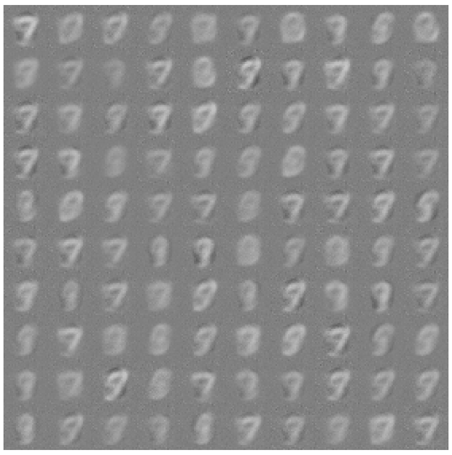
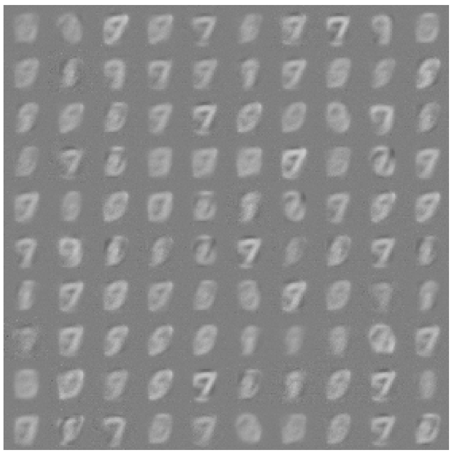
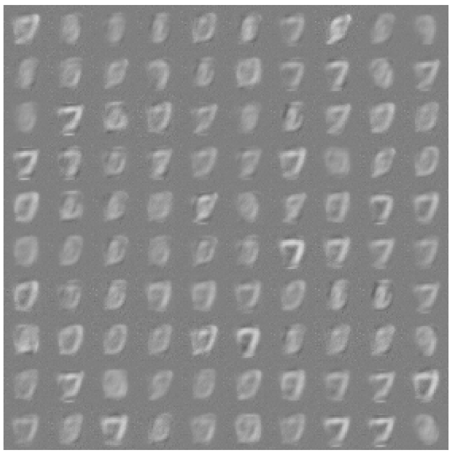


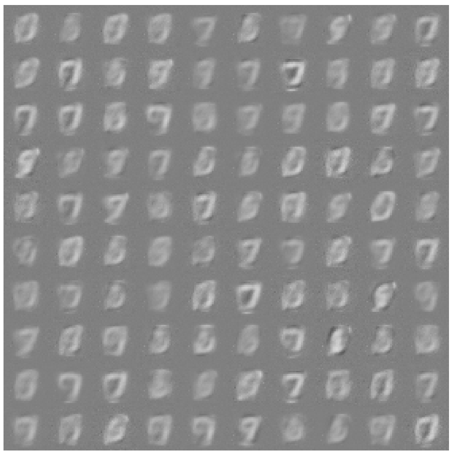
Figure 3.3: The corresponding images after training


(a) epoch=1


(b) epoch=3


(c) epoch=5


(d) epoch=7


(e) epoch=8

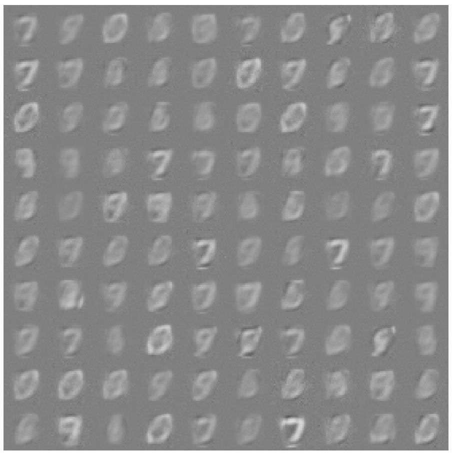

(f) epoch=9

Figure 3.4: The corresponding images for some epochs