



به نام خدا



دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

مدل‌های مولڈ عمیق

تمرین اول

محمد طاها مجلسی کوپایی

نام و نام خانوادگی

۸۱۰۱۰۱۵۰۴

شماره دانشجویی

۱۴۰۳/۸/۱۴

تاریخ ارسال گزارش

فهرست

۳	سوال اول زیر بخش اول
۵	سوال اول زیر بخش دوم
۶	سوال اول زیر بخش سوم
۹	سوال اول زیر بخش چهارم
۱۰	سوال اول زیر بخش پنجم
۱۳	بخش دوم سوال اول
۱۶	بخش دوم سوال دوم
۱۸	بخش دوم سوال سوم
۲۷	بخش دوم تئوری سوال اول
۳۲	بخش دوم تئوری سوال دوم
۳۷	بخش دوم تئوری سوال سوم
۴۲	مراجع

بخش اول - PGM

زیربخش اول :

رسم شبکه بیزی:

با توجه به توضیحات و متغیرهای داده شده، شبکه بیزی به صورت زیر رسم می‌شود:

• متغیرها:

- A: سیستم تهویه هوا دچار نقص شود.
- O: در اتاق سرور برای مدت طولانی باز بماند.
- T: دمای اتاق از یک حد آستانه بیشتر شود.
- M: پیامی متنی دریافت شود.
- S: آذیر اخطار به صدا در بیاید.

• روابط بین متغیرها:

- A و O تأثیر مستقیم بر T دارند.
- يعني A و O والدین T هستند.
- T تأثیر مستقیم بر M و S دارد.
- يعني T والد M و S است.

شبکه بیزی به این صورت می‌باشد: (به دلیل محدودیت صرفاً یال‌ها را رسم کردیم)

$$A \rightarrow T \leftarrow O$$

$$T \rightarrow M \quad T \rightarrow S$$

توزيع احتمال توأم:

با توجه به ساختار شبکه بیزی، توزیع احتمال توأم به صورت زیر بیان می‌شود:

$$P(A, O, T, M, S) = P(A) \times P(O) \times P(T|A, O) \times P(M|T) \times P(S|M)$$

بررسی صحت عبارات:

$$O \perp A \quad (3.1)$$

پاسخ: درست است.

- در شبکه بیزی، O و A هیچ یالی بین خود ندارند و والدین مستقل هستند. و هم چنین فزند مشترک داده نشده.
- بنابراین، O و A مستقل هستند.

$$: O \perp A | M \quad (3.2)$$

پاسخ: نادرست است.

- وقتی روی M شرط می‌گیریم، که یک فرزند T است، وابستگی بین O و A ایجاد می‌شود.
- زیرا O و A هر دو بر T تأثیر می‌گذارند، و T نیز بر M تأثیر می‌گذارد.
- در این حالت، با دانستن M ، اطلاعاتی درباره T کسب می‌کنیم که می‌تواند وابستگی بین O و A را ایجاد کند.

$$S \perp M \quad (3.3)$$

پاسخ: نادرست است.

- S و M هر دو فرزندان مستقیم T هستند.
- بدون شرطگیری روی T ، S و M به واسطه T وابسته هستند.

- بنابراین، S و M مستقل نیستند.

$$S \perp M | O \quad (3.4)$$

پاسخ: نادرست است.

- شرطگیری روی O وابستگی بین S و M را از بین نمیبرد.
- زیرا مسیر وابستگی از S به M از طریق T همچنان وجود دارد.
- برای مستقل شدن S و M ، باید روی T شرط بگیریم، نه O .

جمع‌بندی:

- (3.1) درست: O و A مستقل هستند.
- (3.2) نادرست: O و A با دانستن M وابسته می‌شوند.
- (3.3) نادرست: S و M وابسته هستند.
- (3.4) نادرست: شرطگیری روی O وابستگی بین S و M را از بین نمیبرد.

سؤال اول

زیربخش دوم :

برای پاسخ به این سوال، ابتدا باید به ساختار و وابستگی‌های موجود در گراف مارکوف توجه کنیم. گراف مارکوف، یک ساختار وابستگی شرطی است که در آن هر گره (متغیر) تنها به گره‌های مجاور خود وابسته است. بنابراین، با استفاده از قوانین استقلال شرطی در گراف مارکوف می‌توان درستی یا نادرستی هر یک از عبارات داده شده را بررسی کرد.

۱. $I \perp F$: این عبارت بیان می‌کند که A و F مستقل هستند. با توجه به ساختار گراف، A و F به واسطه G متصل شده‌اند، بنابراین مستقل نیستند. این عبارت نادرست است.

۲. $B \perp I|F$: این عبارت بیان می‌کند که B و A تحت شرط F مستقل هستند. از آنجا که B و A در گراف به واسطه زنجیره‌ای از گره‌ها متصل شده‌اند و شرط F این زنجیره را قطع نمی‌کند، این عبارت نادرست است. بنابراین، این عبارت نادرست است.

۳. $A \perp G|C, E$: این عبارت بیان می‌کند که A و G تحت شرط C و E مستقل هستند. با توجه به گراف، A و G با زنجیره‌ای از گره‌ها به هم وصل هستند ولی با وجود C و E به هم مسیر مستقیم ای پیدا نمی‌کند. بنابراین، این عبارت نادرست است.

۴. $P(B|C, D, F) = P(B|C, D)$: این عبارت بیان می‌کند که احتمال B تحت شرط C و D برابر با احتمال B تحت شرط C و D است. با توجه به ساختار گراف و اینکه F تأثیری روی B در حضور C و D ندارد، این استقلال شرطی برقرار است. بنابراین، این عبارت درست است. پس :

- عبارت ۱ نادرست است.
- عبارت ۲ نادرست است.
- عبارت ۳ نادرست است.
- عبارت ۴ درست است.

سؤال اول

زیر بخش سوم :

۱. توزیع توام متغیرها با توجه به گراف بیزی: برای نوشتن توزیع توام با توجه به گراف بیزی داده شده، از فرم فاکتوریزه استفاده می‌کنیم. اگر این گراف بیزی را با توجه به گره‌ها و یال‌های آن تحلیل کنیم، توزیع توام به صورت زیر فاکتوریزه می‌شود:

$$P(X_1, X_2, X_3, X_4, X_5, X_6, X_7) = P(X_1) \times P(X_2|X_1) \times P(X_3|X_1) \times P(X_4|X_2) \times P(X_5|X_2, X_3) \times P(X_6|X_5) \times P(X_7|X_6)$$

۲. X_6 مربوط به متغیر X_6 blanket مارکوف یک متغیر، مجموعه‌ای از متغیرهای است که با شرط‌بندی بر روی آنها، متغیر مورد نظر نسبت به سایر متغیرهای گراف مستقل می‌شود. برای X_6 blanket مارکوف شامل والدین (X_5 ، فرزندان (X_7) و والدین فرزندان (در اینجا، متغیر X_4 این گونه است). می‌شود. بنابراین: $X_5 X_7 | X_4$ markov Blanket برابر با خواهد شد و

۳. رسم گراف مارکوف مربوط به گراف بیزی: دقیقاً یال‌های قبلى باقی خواهند ماند و بدون جهت می‌شوند ولی یال‌های دیگری هم به خاطر ساختار V-structure اضافه خواهند شد.

X_4-X_6 X_2-X_3

۴. آیا گراف بستی آمده در قسمت ۳ perfect i-map مربوط به گراف بیزی است؟ چرا؟ خیر چون بعضی از استقلال‌هایی که در گراف اصلی وجود داشت در گراف تبدیل یافته وجود ندارد. مثلاً در یک V-structure اگر فرزند را داشته باشیم دو پدر از هم مسقل خواهند شد. اما در اینجا این گونه نیست. و به طور مثال این نوع از استقلال را نداشتیم.

۵. آیا گراف بستی آمده در قسمت ۳ chordal است؟ چرا؟ بله کورDAL می‌باشد چون در هر دور به طول حداقل ۴ یک یال داریم برای دو راس مجاور از هم. یا به نحو دیگر گراف مثلثی می‌باشد پس کورDAL است.

تعريف گراف chordal

گرافی را **chordal** می‌گوییم (گاهی هم گراف بدون چرخه القایی یا گراف بدون حفره نامیده می‌شود) اگر هر چرخه‌ای که طول آن ۴ یا بیشتر است، دارای یالی بین دو گره غیرمتوالی باشد. به عبارت دیگر، در یک گراف **chordal**، هر چرخه‌ای که حداقل چهار گره داشته باشد، باید یک میانبر (**chord**) داشته باشد که آن چرخه را به مسیرهای کوتاه‌تر تقسیم کند.

ویژگی اصلی گرافهای **chordal**

ویژگی اصلی گرافهای **chordal** این است که آنها از چرخه‌های بلند بدون میانبر، یا به عبارت دیگر چرخه‌های بدون **chord**، تشکیل نشده‌اند. برای مثال:

- یک چرخه به طول ۴ (چهار گره که به صورت پشت سر هم با یال به هم متصل شده‌اند) باید حداقل یک یال اضافی بین دو گره غیرمتوالی داشته باشد تا بتوان آن را به عنوان یک گراف **chordal** در نظر گرفت.
- اگر گراف شامل چرخه‌ای با طول ۵ یا بیشتر باشد و در این چرخه هیچ یال اضافی وجود نداشته باشد، این گراف **chordal** نیست.

چرا گراف داده شده در سوال **chordal** نیست؟

در قسمت سوال، گرافی که به عنوان **بستی مارکوف** از گراف بیزی ساخته شده است، ممکن است شامل چرخه‌هایی باشد که در آنها میانبر (**chord**) وجود ندارد. به عنوان مثال، اگر در گراف مارکوف چرخه‌ای به طول ۴ یا بیشتر باشد و این چرخه بدون یال اضافی بین گره‌های غیرمتوالی باشد، این چرخه خاصیت **chordal** بودن را نقض می‌کند.

مثال ساده

فرض کنید یک چرخه ساده با چهار گره A B C و D داریم که به صورت زیر به هم متصل شده‌اند:

$$A - B - C - D - A$$

در این چرخه، هیچ یالی بین A و C یا بین B و D وجود ندارد. بنابراین این چرخه دارای طول ۴ بدون **chord** است و باعث می‌شود که گراف **غیر-chordal** باشد. اگر بخواهیم این گراف را **chordal** کنیم، باید یکی از یال‌های A به C یا B به D را اضافه کنیم.

سؤال اول

زیر بخش چهارم:

1. اگر G یک گراف بیزی باشد: یک گراف بیزی نمایانگر توزیع احتمال توام بر روی مجموعه‌ای از متغیرهای استقلال های شرطی خاصی را نشان دهد. اگر G یک perfect i-map باشد، به این معنی است که تمام استقلال های شرطی موجود در توزیع، توسط ساختار گراف بیزی G به طور دقیق نشان داده می‌شوند. اگر از G یالی را حذف کنیم و گراف جدید G' را بسازیم، در این صورت ممکن است برخی از استقلال های شرطی جدید در G' ایجاد شوند که قبلاً در G وجود نداشته‌اند. بنابراین، G' نمی‌تواند perfect i-map برای همان لیست استقلال L باشد.
2. اگر G یک گراف مارکوف باشد: گراف مارکوف نیز لیست استقلال‌ها را به صورت شرطی نشان می‌دهد. در این حالت هم، اگر G یک perfect i-map برای لیست استقلال L باشد، با حذف یالی از G و ساختن G' ممکن است استقلال های جدیدی ایجاد شوند که لیست استقلال L را به درستی نشان ندهند. در نتیجه، G' نمی‌تواند perfect i-map برای L باشد.

سؤال اول

زیر بخش پنجم:

برای تخمین توزیع $p(z_1|x_1, x_2)$ با استفاده از تقریب لاپلاس، مراحل زیر را دنبال می‌کنیم:

۱. نوشتن احتمال مشترک:

احتمال مشترک را می‌توان به صورت زیر بیان کرد:

$$p(z_1, x_1, x_2) = p(z_1) p(x_1|z_1) p(x_2|z_1)$$

با توجه به توزیع‌های داده شده:

$$p(z_1) = N(0, 1), \quad p(x_i|z_1) = N(x_i|z_1, \sigma_i^2) \quad \text{for } i = 1, 2$$

۲. محاسبه لگاریتم پسین (تا یک ثابت):

$$\ln p(z_1|x_1, x_2) = \ln p(z_1) + \ln p(x_1|z_1) + \ln p(x_2|z_1) + \text{const}$$

با جایگذاری توزیع‌های نرمال و ساده‌سازی:

$$-\ln p(z_1|x_1, x_2) = \frac{1}{2} z_1^2 + \frac{1}{2\sigma_1^2} (x_1 - z_1)^2 + \frac{1}{2\sigma_2^2} (x_2 - z_1)^2 + \text{const}$$

۳. یافتن مد Z با مشتق‌گیری:

مشتق را نسبت به z_1 محاسبه کرده و برابر صفر قرار می‌دهیم:

$$\frac{d}{dz_1} \left(\frac{1}{2} z_1^2 + \frac{1}{2\sigma_1^2} (x_1 - z_1)^2 + \frac{1}{2\sigma_2^2} (x_2 - z_1)^2 \right) = 0$$

ساده‌سازی مشتق:

$$z_1 - \frac{(x_1 - z_1)}{\sigma_1^2} - \frac{(x_2 - z_1)}{\sigma_2^2} = 0$$

بازنویسی و حل برای \hat{z}_1 :

$$z_1 \left(1 + \frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2} \right) = \frac{x_1}{\sigma_1^2} + \frac{x_2}{\sigma_2^2}$$

$$\hat{z}_1 = \frac{\frac{x_1}{\sigma_1^2} + \frac{x_2}{\sigma_2^2}}{1 + \frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}}$$

۴. محاسبه واریانس $\hat{\sigma}^2$

مشتق دوم منفی لگاریتم پسین، معکوس واریانس را می‌دهد:

$$\frac{d^2}{dz_1^2}(\dots) = 1 + \frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}$$

$$\hat{\sigma}^2 = \left(1 + \frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2} \right)^{-1}$$

۵. نوشتن توزیع پسین تقریبی:

توزیع پسین تقریبی یک توزیع نرمال با میانگین \hat{z}_1 و واریانس $\hat{\sigma}^2$ است:

$$q(z_1) = N(z_1 \mid \hat{z}_1, \hat{\sigma}^2)$$

پاسخ نهایی:

تقریب نرمال به صورت زیر است:

$$q(z_1) = N \left(z_1 \mid \frac{\frac{x_1}{\sigma_1^2} + \frac{x_2}{\sigma_2^2}}{1 + \frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}}, \quad \left(1 + \frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2} \right)^{-1} \right)$$

بخش دوم : VAE

سوال اول:

در مدل‌های خودرمزگذار تنوعی (VAE)، محاسبه مستقیم تابع درست‌نمایی داده‌ها $\log p(\mathbf{x})$ به دلیل پیچیدگی محاسباتی بسیار دشوار است. به منظور حل این مشکل، از کران پایین شواهد (ELBO) استفاده می‌کنیم که محاسبه آن ساده‌تر است. هدف ما بیشینه کردن ELBO است که به طور

غیرمستقیم به بیشینه کردن $\log p(\mathbf{x})$ کمک می‌کند.

تعریف ELBO

$$\text{ELBO} = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{x}|\mathbf{z})] - \text{KL}(q(\mathbf{z}|\mathbf{x}) || p(\mathbf{z}))$$

• اصطلاح بازسازی داده‌ها، نشان‌دهنده کیفیت بازسازی X از Z $E_{q(z|x)}[\log p(x|z)]$ است.

• $KL(q(z|x) || p(z))$: فاصله کولبک-لیبلر بین توزیع نهان تقریبی $q(z|x)$ و توزیع پیشین $p(z)$ است.

اثبات رابطه بین ELBO و $\log p(\mathbf{x})$

هدف ما نشان دادن این رابطه است:

$$\log p(\mathbf{x}) = \text{ELBO} + \text{KL}(q(\mathbf{z}|\mathbf{x}) || p(\mathbf{z}|\mathbf{x}))$$

که در آن $p(z|x)$ توزیع پسین واقعی است و $q(z|x)$ توزیع پسین تقریبی است.

مراحل اثبات:

1. شروع از تعریف :

$$\log p(\mathbf{x}) = \log \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z}$$

2. معرفی توزیع :

$$\log p(\mathbf{x}) = \log \int q(\mathbf{z}|\mathbf{x}) \frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z}|\mathbf{x})} d\mathbf{z}$$

3. استفاده از نابرابری جنسن:

$$\log p(\mathbf{x}) \geq \int q(\mathbf{z}|\mathbf{x}) \log \left(\frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z}|\mathbf{x})} \right) d\mathbf{z}$$

4. تقسیم به دو قسمت:

$$\log p(\mathbf{x}, \mathbf{z}) = \log p(\mathbf{x}|\mathbf{z}) + \log p(\mathbf{z})$$

5. جایگذاری و سادهسازی:

$$\log p(\mathbf{x}) \geq \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{x}|\mathbf{z}) + \log p(\mathbf{z}) - \log q(\mathbf{z}|\mathbf{x})]$$

$$\log p(\mathbf{x}) \geq \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{x}|\mathbf{z})] - \text{KL}(q(\mathbf{z}|\mathbf{x}) || p(\mathbf{z}))$$

این همان تعریف ELBO است:

$$\text{ELBO} = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{x}|\mathbf{z})] - \text{KL}(q(\mathbf{z}|\mathbf{x}) || p(\mathbf{z}))$$

6. تعریف تفاوت بین ELBO و :

$$\log p(\mathbf{x}) - \text{ELBO} = \text{KL}(q(\mathbf{z}|\mathbf{x}) || p(\mathbf{z}|\mathbf{x}))$$

: زیرا

$$\text{KL}(q(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z}|\mathbf{x})) = \log p(\mathbf{x}) - \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{x}|\mathbf{z}) + \log p(\mathbf{z}) - \log q(\mathbf{z}|\mathbf{x})]$$

که پس از ساده‌سازی برابر است با:

$$\text{KL}(q(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z}|\mathbf{x})) = \log p(\mathbf{x}) - \text{ELBO}$$

نتیجه‌گیری:

- رابطه نهایی:

$$\log p(\mathbf{x}) = \text{ELBO} + \text{KL}(q(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z}|\mathbf{x}))$$

- تفسیر:

- همیشه مقداری غیرمنفی است.

- بنابراین، ELBO یک کران پایین برای $\log p(\mathbf{x})$ است.

- با بیشینه کردن ELBO، ما به طور غیرمستقیم $\text{KL}(q(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z}|\mathbf{x}))$ را کمینه می‌کنیم.

- این باعث می‌شود که توزیع تقریبی $q(\mathbf{z}|\mathbf{x})$ به توزیع پسین واقعی $p(\mathbf{z}|\mathbf{x})$ نزدیک‌تر شود.

بخش دوم : VAE

سوال دوم :

در آموزش یک خودرمزگذار تنووعی (VAE)، هدف ما بیشینه کردن کران پایین شواهد (ELBO) است.

این شامل نمونهگیری از توزیع پسین تقریبی $q(\mathbf{z}|\mathbf{x})$ و محاسبه گرادیان‌ها نسبت به پارامترهای مدل منشود. با این حال، نمونهگیری مستقیم از $q(\mathbf{z}|\mathbf{x})$ استوکاستیسیته را به گراف محاسباتی وارد می‌کند، که پساننتشار (Backpropagation) را غیرممکن می‌سازد زیرا نمی‌توانیم گرادیان‌ها را از طریق عملیات نمونهگیری تصادفی محاسبه کنیم.

برای حل این مشکل، از ترفند بازپارامتری‌سازی استفاده می‌کنیم. این تکنیک عملیات نمونهگیری تصادفی را به یک عملیات قطعی تبدیل می‌کند، که اجازه می‌دهد گرادیان‌ها از طریق پساننتشار محاسبه شوند.

ترفند بازپارامتری‌سازی:

1. تبدیل نمونهگیری:

به جای نمونهگیری مستقیم \mathbf{z} از $q(\mathbf{z}|\mathbf{x})$ را به صورت یکتابع قطعی از \mathbf{x} و یک متغیر نویز

تصادفی ϵ بیان می‌کنیم:

$$\mathbf{z} = \mu(\mathbf{x}) + \sigma(\mathbf{x}) \odot \epsilon$$

که در آن:

$\mu(\mathbf{x})$ و $\sigma(\mathbf{x})$ خروجی‌های شبکه کدگذار هستند (توابعی از \mathbf{x}).

$\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ یک بردار نویز تصادفی نمونهگیری شده از توزیع نرمال استاندارد است.

○ \odot نشان‌دهنده ضرب عنصر به عنصر است.

2. تابع قطعی:

تبديل $\mathbf{z} = g(\epsilon, \mathbf{x})$ اکنون یک تابع قطعی نسبت به پارامترهای مدل است (جاسازی شده در ELBO را نسبت به پارامترها با استفاده از $\mu(\mathbf{x})$ و $\sigma(\mathbf{x})$. این به ما اجازه می‌دهد تا گرادیان‌های ELBO را پس انتشار استاندارد محاسبه کنیم.

3. محاسبه ELBO:

به صورت زیر است:

$$\mathcal{L}(\mathbf{x}) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{x}|\mathbf{z})] - \text{KL}(q(\mathbf{z}|\mathbf{x}) || p(\mathbf{z}))$$

4. با استفاده از ترفندهای بازپارامتری‌سازی، امید ریاضی را می‌توان به صورت زیر تقریب زد:

$$\mathcal{L}(\mathbf{x}) \approx \frac{1}{L} \sum_{l=1}^L \left[\log p(\mathbf{x}|\mathbf{z}^{(l)}) \right] - \text{KL}(q(\mathbf{z}|\mathbf{x}) || p(\mathbf{z}))$$

$\mathbf{z}^{(l)} = \mu(\mathbf{x}) + \sigma(\mathbf{x}) \odot \epsilon^{(l)}$ که در آن L تعداد نمونه‌ها است.

بخش دوم : VAE

بخش ج :

مدل ساخته شده مطابق با صورت سوال به این شکل میباشد :

```
class VariationalAutoencoder(nn.Module):
    def __init__(self, hidden_dim=4096, latent_dim=32):
        super(VariationalAutoencoder, self).__init__()

        self.encoder_net = nn.Sequential(
            nn.Conv2d(in_channels=3, out_channels=32, kernel_size=4, stride=2, padding=1),
            # nn.BatchNorm2d(32),
            nn.LeakyReLU(0.2),
            # nn.Dropout(0.2),

            nn.Conv2d(in_channels=32, out_channels=64, kernel_size=4, stride=2, padding=1),
            nn.BatchNorm2d(64),
            nn.LeakyReLU(0.2),
            nn.Dropout(0.2),

            nn.Conv2d(in_channels=64, out_channels=128, kernel_size=4, stride=2, padding=1),
            nn.BatchNorm2d(128),
            nn.LeakyReLU(0.2),
            nn.Dropout(0.3),

            nn.Conv2d(in_channels=128, out_channels=256, kernel_size=4, stride=2, padding=1),
            # nn.BatchNorm2d(256),
            nn.LeakyReLU(0.2),
            # nn.Dropout(0.3),

            nn.Flatten(),
            nn.Linear(in_features=256*8*8, out_features=hidden_dim),
            # nn.BatchNorm1d(hidden_dim),
            nn.LeakyReLU(0.2),
            # nn.Dropout(0.3),
        )

        self.mean_fc = nn.Linear(hidden_dim, latent_dim)
        self.logvar_fc = nn.Linear(hidden_dim, latent_dim)

        self.decoder_input_fc = nn.Linear(latent_dim, hidden_dim)
        self.decoder_net = nn.Sequential(
            # nn.BatchNorm1d(hidden_dim),
            nn.LeakyReLU(0.2),
            nn.Linear(hidden_dim, 256*8*8),
            nn.LeakyReLU(0.2),
            # nn.Dropout(0.3),

            nn.Unflatten(dim=1, unflattened_size=(256, 8, 8)),

            nn.ConvTranspose2d(in_channels=256, out_channels=128, kernel_size=4, stride=2, padding=1),
            nn.BatchNorm2d(128),
            nn.LeakyReLU(0.2),
            nn.Dropout(0.3),

            nn.ConvTranspose2d(in_channels=128, out_channels=64, kernel_size=4, stride=2, padding=1),
            nn.BatchNorm2d(64),
            nn.LeakyReLU(0.2),
            nn.Dropout(0.2),

            nn.ConvTranspose2d(in_channels=64, out_channels=32, kernel_size=4, stride=2, padding=1),
            # nn.BatchNorm2d(32),
            nn.LeakyReLU(0.2),
            # nn.Dropout(0.2),

            nn.ConvTranspose2d(in_channels=32, out_channels=3, kernel_size=4, stride=2, padding=1),
            nn.Tanh(),
        )
```

از جهت آن که قدرت پردازشی زیادی در دسترس نبود مدل را کمی تغییر دادیم تا به نتایج بهتری رسیده باشیم.

اطلاعات اولیه برای train شدن :

```
num_epochs = 1000
initial_learning_rate = 0.0005

vae_model = VariationalAutoencoder().to(device)
optimizer = torch.optim.Adam(vae_model.parameters(), lr=initial_learning_rate)

train_total_losses = []
validation_losses = []
train_reconstruction_losses = []
train_kl_divergence_losses = []
```

ساختار کد بخش train مدل به این صورت بود که صرفا مدل را بر اساس تابع هزینه‌ی مربوطه آپدیت کرده بود:

```

for epoch in range(num_epochs):
    vae_model.train()
    epoch_train_loss = 0
    epoch_reconstruction_loss = 0
    epoch_kl_divergence_loss = 0

    for batch_index, (batch_data, _) in enumerate(training_data_loader):
        batch_data = batch_data.to(device)
        optimizer.zero_grad()

        reconstructed_batch, latent_mu, latent_logvar = vae_model(batch_data)
        total_loss, reconstruction_loss, kl_divergence_loss = vae_loss_function(
            reconstructed_batch, batch_data, latent_mu, latent_logvar
        )

        total_loss.backward()
        optimizer.step()

        epoch_train_loss += total_loss.item()
        epoch_reconstruction_loss += reconstruction_loss.item()
        epoch_kl_divergence_loss += kl_divergence_loss.item()

    train_total_losses.append(epoch_train_loss / len(training_data_loader.dataset))
    train_reconstruction_losses.append(epoch_reconstruction_loss / len(training_data_loader.dataset))
    train_kl_divergence_losses.append(epoch_kl_divergence_loss / len(training_data_loader.dataset))

    vae_model.eval()
    epoch_validation_loss = 0

    with torch.no_grad():
        for val_data, _ in validation_data_loader:
            val_data = val_data.to(device)
            recon_val_batch, val_mu, val_logvar = vae_model(val_data)
            val_loss, _, _ = vae_loss_function(recon_val_batch, val_data, val_mu, val_logvar)
            epoch_validation_loss += val_loss.item()

    validation_losses.append(epoch_validation_loss / len(validation_data_loader.dataset))

    print(f'Epoch {epoch}, Train Loss: {train_total_losses[-1]:.4f}, Validation Loss: {validation_losses[-1]:.4f}')

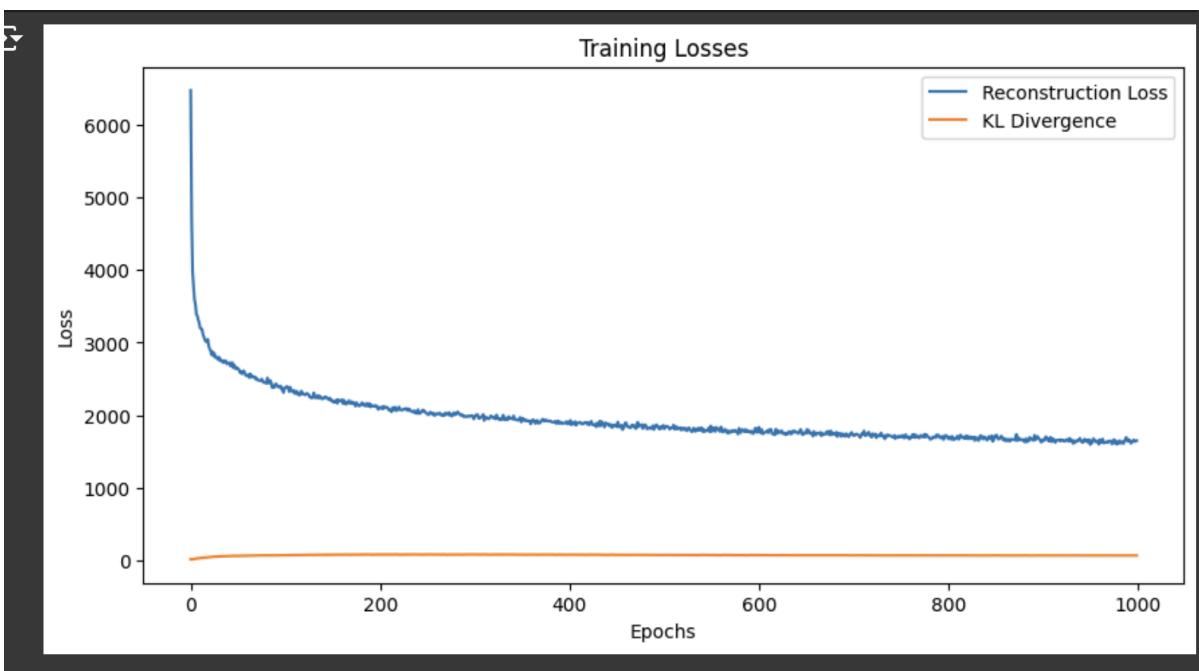
```

در این کد، فرآیند آموزش و ارزیابی مدل خودرمزنگذار متغیر (VAE) به طور کامل پیاده‌سازی شده است. هدف این فرآیند آموزش مدل برای بازسازی داده‌ها به شکلی است که هم خطای بازسازی کاهش یابد و هم فضای نهان منظم و فشرده‌ای یاد گرفته شود.

وضعیت epoch های پایانی در مدل :

```
2m └── Epoch 961, Train Loss: 1712.5202, Validation Loss: 1949.6487
    └── Epoch 962, Train Loss: 1692.0791, Validation Loss: 1967.5531
        └── Epoch 963, Train Loss: 1743.7827, Validation Loss: 1940.3621
            └── Epoch 964, Train Loss: 1691.6240, Validation Loss: 1938.0363
                └── Epoch 965, Train Loss: 1691.2165, Validation Loss: 1998.0003
                    └── Epoch 966, Train Loss: 1700.1985, Validation Loss: 1925.8239
                        └── Epoch 967, Train Loss: 1697.4328, Validation Loss: 1936.5205
                            └── Epoch 968, Train Loss: 1708.6008, Validation Loss: 1965.4723
                                └── Epoch 969, Train Loss: 1698.2088, Validation Loss: 1954.9462
                                    └── Epoch 970, Train Loss: 1711.8245, Validation Loss: 1938.1280
                                        └── Epoch 971, Train Loss: 1718.4156, Validation Loss: 1971.8726
                                            └── Epoch 972, Train Loss: 1709.8836, Validation Loss: 1942.8837
                                                └── Epoch 973, Train Loss: 1674.0670, Validation Loss: 1940.3383
                                                    └── Epoch 974, Train Loss: 1706.9505, Validation Loss: 1936.4837
                                                        └── Epoch 975, Train Loss: 1735.8962, Validation Loss: 1945.9228
                                                            └── Epoch 976, Train Loss: 1722.3701, Validation Loss: 1994.9492
                                                                └── Epoch 977, Train Loss: 1673.2198, Validation Loss: 1936.0148
                                                                    └── Epoch 978, Train Loss: 1710.9364, Validation Loss: 1973.5153
                                                                        └── Epoch 979, Train Loss: 1673.2449, Validation Loss: 1923.6583
                                └── Epoch 980, Train Loss: 1699.6863, Validation Loss: 1937.6884
                                    └── Epoch 981, Train Loss: 1720.3276, Validation Loss: 1948.9264
                                        └── Epoch 982, Train Loss: 1685.5296, Validation Loss: 1943.0112
                                            └── Epoch 983, Train Loss: 1696.6006, Validation Loss: 1949.8939
                                                └── Epoch 984, Train Loss: 1707.3807, Validation Loss: 1914.1065
                                                    └── Epoch 985, Train Loss: 1679.9964, Validation Loss: 1931.7288
                                                        └── Epoch 986, Train Loss: 1714.7194, Validation Loss: 1935.9698
                                                            └── Epoch 987, Train Loss: 1735.0827, Validation Loss: 1956.1084
                                                                └── Epoch 988, Train Loss: 1766.4708, Validation Loss: 1953.9749
                                                                    └── Epoch 989, Train Loss: 1719.3289, Validation Loss: 1951.0376
                                                                        └── Epoch 990, Train Loss: 1709.2564, Validation Loss: 1950.4423
                                └── Epoch 991, Train Loss: 1734.2256, Validation Loss: 1919.4284
                                    └── Epoch 992, Train Loss: 1716.5299, Validation Loss: 1941.5617
                                        └── Epoch 993, Train Loss: 1684.3253, Validation Loss: 1943.4446
                                            └── Epoch 994, Train Loss: 1705.7880, Validation Loss: 2025.8981
                                                └── Epoch 995, Train Loss: 1712.1634, Validation Loss: 1946.8408
                                                    └── Epoch 996, Train Loss: 1732.8812, Validation Loss: 1976.8728
                                                        └── Epoch 997, Train Loss: 1707.9629, Validation Loss: 1936.9383
                                                            └── Epoch 998, Train Loss: 1709.4926, Validation Loss: 1980.3244
                                                                └── Epoch 999, Train Loss: 1721.8064, Validation Loss: 1955.0225
```

: مقدار loss مدل



در این قسمت بعد از تلاش زیاد موفق شدیم هزینه‌ی مدل را به آرامی کاهش بدھیم.

در حالات قبلی که تست شده بود مدل سریعاً روی تصاویر آموزشی اورفیت میکرد که به خاطر نبود ترم‌های نرمالیزیشن در بخش لایه‌های مدل بود که با گذاشتن ترم‌های نرمالایر کردن این مشکلات به طور تقریبی حل شد که نتجه‌ی آن تارشدن تصاویر شده بود.

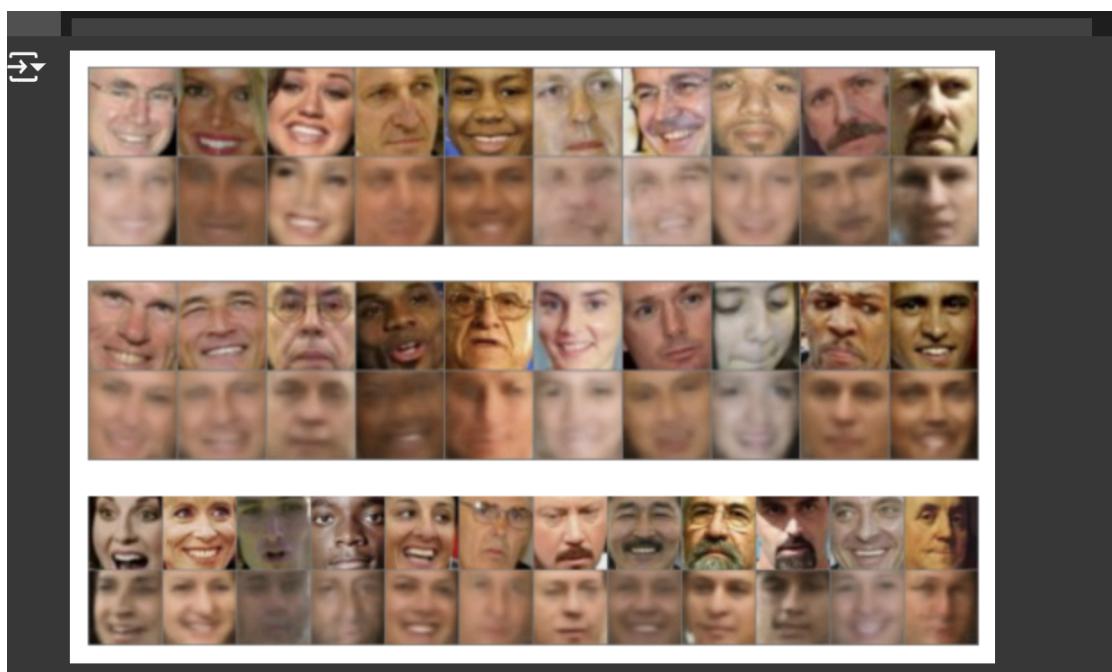
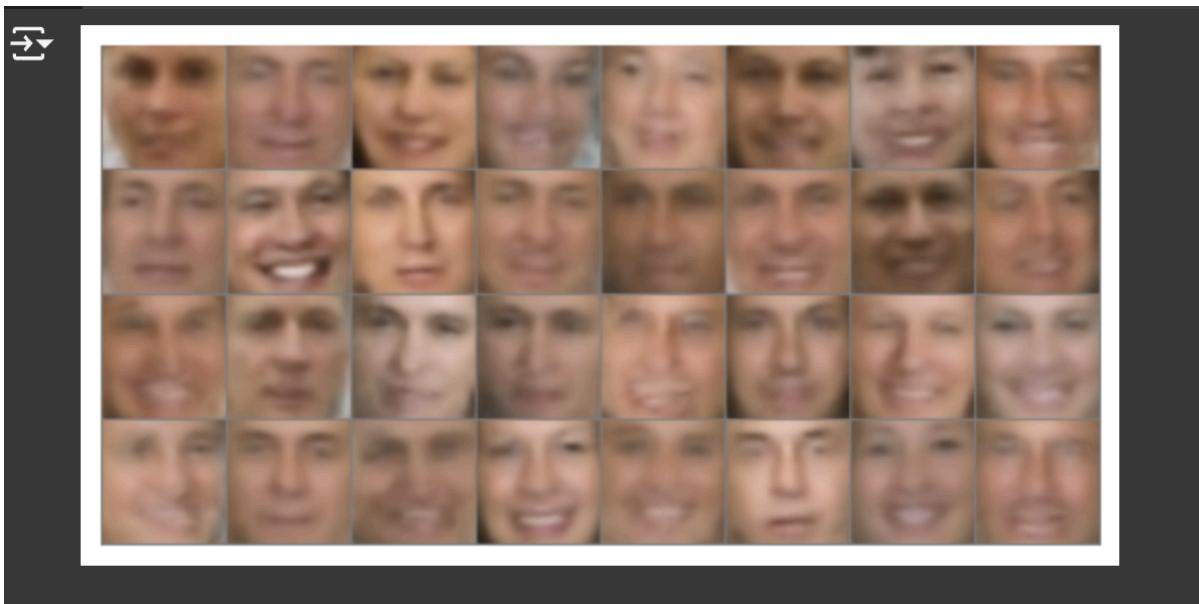
چون در حالت کلی زمان زیادی طول خواهد کشید و دیتای زیادی لازم داریم تا بتوانیم دقیقاً برای هر نقطه در Latent Space یک عکس کامل و با جزئیات کامل تولید کنیم به همین خاطر تصاویر کمی نویزی شده‌اند.

همان‌طور که میبینیم در پایان مدل در تصاویر ۱۲۸ در ۱۲۸ به مقدار لاس ۱۷۰۰ رسیده که یعنی مقدار کمی با مقدار لاس مطوب طراح پروژه که ۱۵۰۰ بود تفاوت دارد.

بخش دوم :

بخش د :

تصاویر تولید شده از مدل بود در صورتی که ورودی تصاویری را به مدل ۱۲۸ در ۱۲۸ میگیرفتیم:



همان طور که میبینیم تصاویر کمی نویزی هستند.

در صورتی که میانگین تصاویر با خنده و بی خنده را میانگین بگیریم به این نتایج خواهیم رسید در پایان :

```

    vae_model.eval()
    with torch.no_grad():
        validation_iterator = iter(validation_data_loader)
        sample_data, _ = next(validation_iterator)
        sample_data = sample_data.to(device)

        reconstructed_batch, _, _ = vae_model(sample_data)

        num_images_display = 10

        image_comparison = torch.cat([sample_data[:num_images_display], reconstructed_batch[:num_images_display]])
        image_comparison = image_comparison.cpu()

        grid_image = torchvision.utils.make_grid(image_comparison, nrow=num_images_display)
        display_images(grid_image)

        sample_data, _ = next(validation_iterator)
        sample_data = sample_data.to(device)
        reconstructed_batch, _, _ = vae_model(sample_data)

        num_images_display = 10

        image_comparison = torch.cat([sample_data[:num_images_display], reconstructed_batch[:num_images_display]])
        image_comparison = image_comparison.cpu()

        grid_image = torchvision.utils.make_grid(image_comparison, nrow=num_images_display)
        display_images(grid_image)

    with torch.no_grad():
        sample_data, _ = next(validation_iterator)
        sample_data = sample_data.to(device)

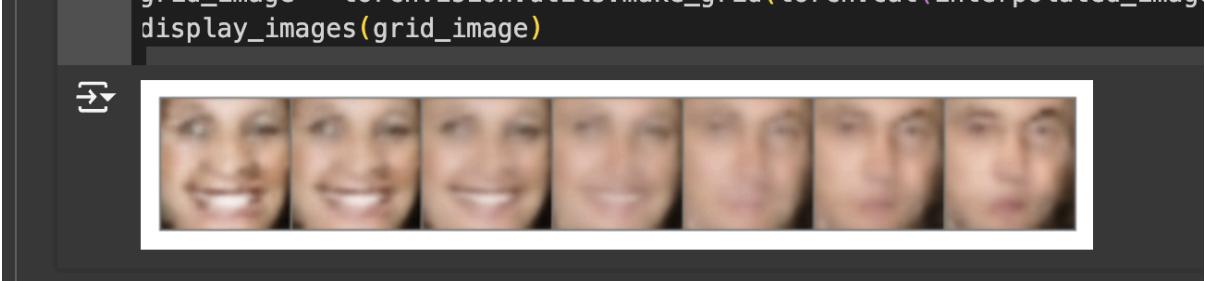
        reconstructed_batch, _, _ = vae_model(sample_data)

        num_images_display = 12

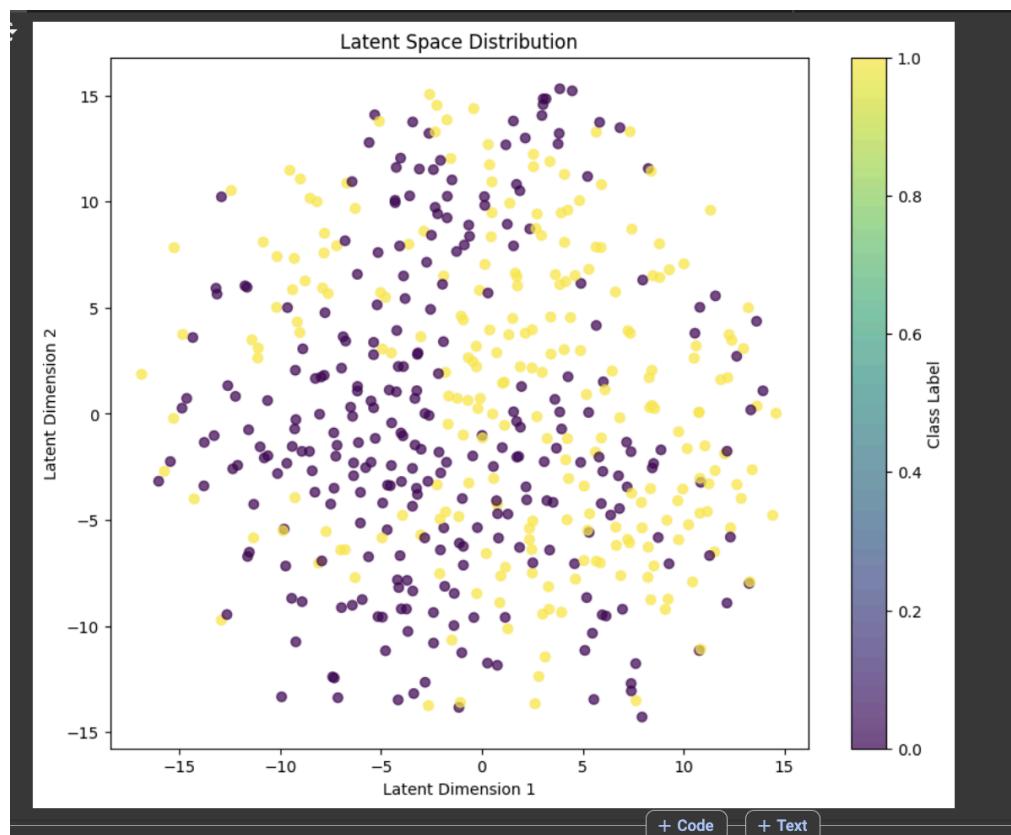
        image_comparison = torch.cat([sample_data[:num_images_display], reconstructed_batch[:num_images_display]])
        image_comparison = image_comparison.cpu()

        grid_image = torchvision.utils.make_grid(image_comparison, nrow=num_images_display)
        display_images(grid_image)

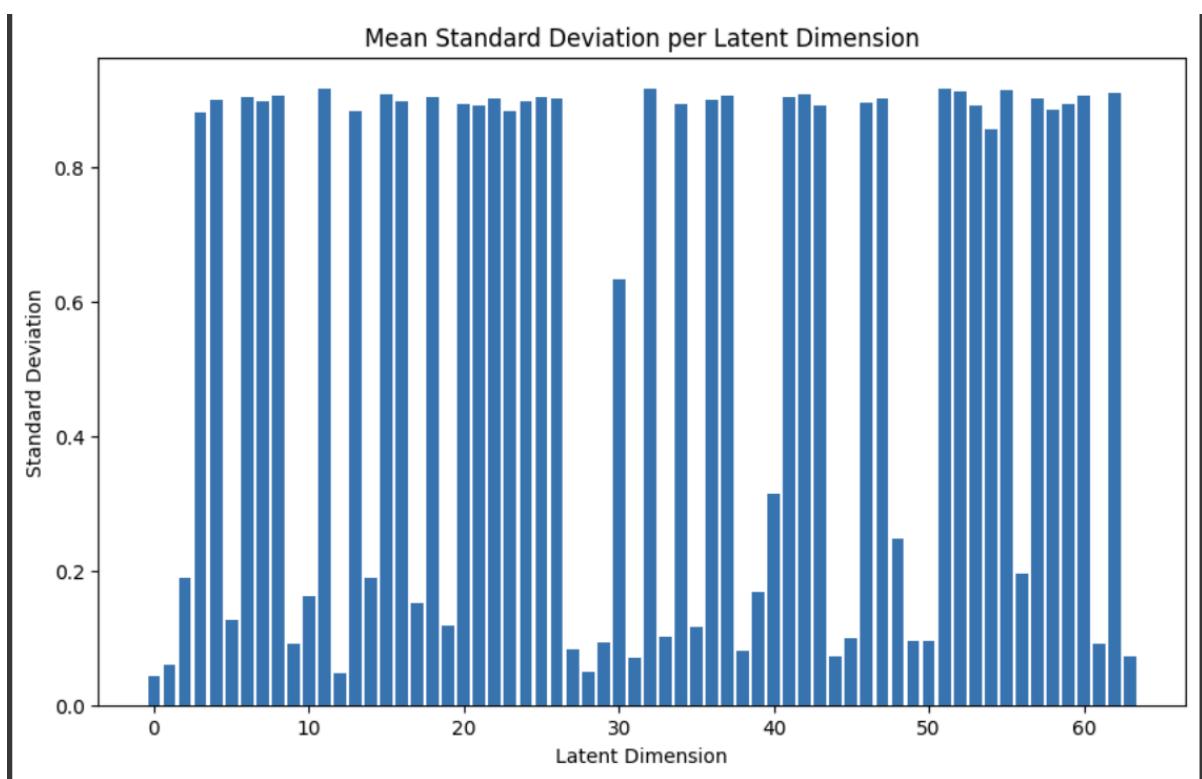
```



و ساختار کد بدین شکل بود که ابتدا باید میانگین تصاویر خنده دار و بی خنده را با استفاده از تابع calculate_latent_means بدست من آوردیم و سپس آن هارا در ۶ جا میانگین وزن دار گرفته و خروجی را بدست من آوردیم همان طور که مشاهده میشود تصاویری حدودا خوب هستند یعنی مدل توانسته فضای میان داده های train را هم تقریب خوبی بزند که به خاطر استفاده از توزیع احتمال ای بود که استفاده کرده بودیم برای بخش میانی . و همین طور میبینیم که فضای latent دارای معنا میباشد .



میانگین واریانس در هر بعد را نشان میدهد:



```

def calculate_latent_means(model, data_loader):
    model.eval()
    latent_means_list, label_collection = [], []
    with torch.no_grad():
        for batch_data, batch_labels in data_loader:
            batch_data = batch_data.to(device)
            encoded_features = model.encoder(batch_data)
            latent_mean = model.mean_fc(encoded_features)
            latent_means_list.append(latent_mean.cpu())
            label_collection.append(batch_labels)
        concatenated_latents = torch.cat(latent_means_list)
        concatenated_labels = torch.cat(label_collection)

        mean_smile_latent = concatenated_latents[concatenated_labels == 0].mean(dim=0)
        mean_nonsmile_latent = concatenated_latents[concatenated_labels == 1].mean(dim=0)

    return mean_smile_latent, mean_nonsmile_latent

```

class Data

```

mean_smile_latent, mean_nonsmile_latent = calculate_latent_means(vae_model, training_data_loader)
latent_direction_vector = (mean_smile_latent - mean_nonsmile_latent).to(device)

validation_data_batch = next(iter(validation_data_loader))
sample_data, sample_labels = validation_data_batch
sample_data = sample_data.to(device)

encoded_sample_data = vae_model.encoder(sample_data)
latent_means_batch = vae_model.mean_fc(encoded_sample_data)

selected_sample_index = 0
z_sample = latent_means_batch[selected_sample_index]

alpha_range = np.linspace(-3, 3, 7)
interpolated_images = []

with torch.no_grad():
    for alpha in alpha_range:
        adjusted_latent = z_sample + alpha * latent_direction_vector
        decoder_input_data = vae_model.decoder_input(adjusted_latent)
        decoded_image = vae_model.decoder(decoder_input_data.unsqueeze(0))
        interpolated_images.append(decoded_image.cpu())

grid_image = torchvision.utils.make_grid(torch.cat(interpolated_images), nrow=len(alpha_range))
display_images(grid_image)

```

این کد یک تابع برای رسم نمودار میانگین انحراف استاندارد برای هر بُعد در فضای پنهان مدل VAE دارد.

1. ابتدا مدل VAE را در حالت ارزیابی قرار می‌دهد و متغیری برای ذخیره انحراف‌های استاندارد ایجاد می‌کند.
2. سپس، از طریق داده‌های ورودی از DataLoader، بدون محاسبه گرادیان، تصاویری را دریافت کرده و آن‌ها را به مدل وارد می‌کند.
3. مدل میانگین و لگاریتم واریانس را برای کدهای پنهان تولید می‌کند. انحراف استاندارد برای هر بُعد پنهان محاسبه و به لیست اضافه می‌شود.
4. داده‌های انحراف استاندارد همه‌ی نمونه‌ها ترکیب و میانگین گرفته می‌شوند.
5. در نهایت، نمودار میله‌ای میانگین انحراف استاندارد برای هر بُعد در فضای پنهان را نمایش می‌دهد.

بخش دوم تئوری :

سوال اول :

مقدمه

در مدل‌های میله‌ای میانگین انحراف Variational Autoencoder - VAE، هدف ما یادگیری توزیع نهان $q(z|x)$ است که بتواند داده‌های واقعی X را به خوبی بازسازی کند. در β -VAE، تأکید بر کشف عوامل نهان جداسده است، به طوری که هر متغیر در بازنمایی نهان تنها به یک عامل خاص حساس باشد و نسبت به سایر عوامل نسبتاً ثابت بماند.

برای دستیابی به این هدف، می‌خواهیم احتمال تولید داده‌های واقعی را بیشینه کنیم، در حالی که فاصله بین توزیع نهان واقعی و تخمین آن را کوچک نگه داریم (مثلاً کمتر از یک ثابت کوچک ۵). این مسئله را می‌توان به صورت یک مسئله بهینه‌سازی با قید فرموله کرد.

فرموله کردن مسئله بهینه‌سازی با قید :

هدف ما بیشینه کردن تابع زیر است:

$$\max_{q(z|x)} \mathbb{E}_{q(z|x)} [\log p(x|z)]$$

با قید:

$$\text{KL}(q(z|x) || p(z)) \leq \delta$$

که در آن $\text{KL}(q(z|x) || p(z))$ واگرایی KL Divergence بین توزیع پسین تقریب $q(z|x)$ و توزیع پیشین $p(z)$ است.

استفاده از لگرانژین و شرایط KKT

برای حل مسئله بهینه‌سازی با قید، از روش لگرانژین و (KKT) استفاده می‌کنیم.

1. ساختن تابع لگرانژین

تابع لگرانژین \mathcal{L} به صورت زیر تعریف می‌شود:

$$\mathcal{L} = -\mathbb{E}_{q(z|x)}[\log p(x|z)] + \lambda (\text{KL}(q(z|x)||p(z)) - \delta)$$

توجه داشته باشید که علامت منفی در جلوی $\mathbb{E}_{q(z|x)}[\log p(x|z)]$ قرار داده شده است، زیرا میخواهیم این عبارت را مینیمم کنیم (برعکس بیشینه‌سازی).

2. شرایط KKT

شرایط KKT برای این مسئله عبارتند از:

: شرایط (a) •

$$\frac{\delta \mathcal{L}}{\delta q(z|x)} = 0$$

: شرط دوگانگی مکمل: •

$$\lambda (\text{KL}(q(z|x)||p(z)) - \delta) = 0$$

: KK شرط (c) •

$$\lambda \geq 0$$

: شرط اولیه قید: •

$$\text{KL}(q(z|x)||p(z)) - \delta \leq 0$$

3. محاسبه مشتق لاغرانژین

: مشتق لاغرانژین نسبت به $q(z|x)$

$$\frac{\delta \mathcal{L}}{\delta q(z|x)} = -\log p(x|z) + \lambda (\log q(z|x) - \log p(z) + 1)$$

4. برابر صفر قرار دادن مشتق لاغرانژین

برای پیدا کردن $q(z|x)$ ، مشتق را برابر صفر قرار می‌دهیم:

$$-\log p(x|z) + \lambda (\log q(z|x) - \log p(z) + 1) = 0$$

5. حل برای $q(z|x)$

بازنویسی معادله:

$$\lambda (\log q(z|x) - \log p(z)) = \log p(x|z) - \lambda$$

سپس:

$$\log q(z|x) = \frac{1}{\lambda} \log p(x|z) + \log p(z) - 1$$

بنابراین:

$$q(z|x) \propto p(x|z)^{\frac{1}{\lambda}} p(z)$$

6. تعریف β

با تعریف $\beta = \lambda^{-1}$ ، می‌توانیم رابطه را ساده کنیم.

7. تابع هزینه نهایی

جایگذاری β در تابع لاغرانژین:

$$\mathcal{L} = -\mathbb{E}_{q(z|x)} [\log p(x|z)] + \beta (\text{KL}(q(z|x)||p(z)) - \delta)$$

از آنجایی که δ یک ثابت است و در فرآیند بهینه‌سازی تأثیری ندارد (چون طبق شرط دوگانگی مکمل)، می‌توانیم آن را نادیده بگیریم.

بنابراین، تابع هزینه به صورت زیر ساده می‌شود:

$$\mathcal{L}_\beta = -\mathbb{E}_{q(z|x)}[\log p(x|z)] + \beta \text{KL}(q(z|x)||p(z))$$

یا با تغییر علامت:

$$\mathcal{L}_\beta = \mathbb{E}_{q(z|x)}[\log p(x|z)] - \beta \text{KL}(q(z|x)||p(z))$$

با استفاده از روش لگرانژین و شرایط KKT، نشان دادیم که با اعمال قید بر روی واگرایی KL بین

$p(z)$ و $q(z|x)$ ، تابع هزینه β -VAE به دست می‌آید:

$$\mathcal{L}_\beta = \mathbb{E}_{q(z|x)}[\log p(x|z)] - \beta \text{KL}(q(z|x)||p(z))$$

توضیحات تکمیلی

- ضریب β نقش مهمی در کنترل تعادل بین بازسازی داده‌ها و جداشده‌گی عوامل نهان دارد. با افزایش β پنالتی بیشتری بر روی واگرایی KL اعمال می‌شود که به جداشده‌گی بیشتر منجر می‌شود، اما ممکن است کیفیت بازسازی را کاهش دهد.
- روش لگرانژین به ما اجازه می‌دهد تا مسئله بهینه‌سازی با قید را به یک مسئله بدون قید تبدیل کنیم که با مینیمم کردن تابع لگرانژین حل می‌شود.
- شرایط KKT تضمین می‌کنند که راه حل بهینه ما قیدهای مسئله را ارضاء می‌کند و در نقطه مینیمم تابع لگرانژین قرار دارد

سؤال دوم تئوری :

بخش دوم

در مدل β -VAE، تابع هزینه به صورت زیر تعریف می‌شود:

$$\mathcal{L}_\beta = \mathbb{E}_{q(z|x)}[\log p(x|z)] - \beta \text{KL}(q(z|x)||p(z))$$

که در آن:

• اصطلاح بازسازی که سعی در ماقسیم کردن احتمال بازسازی $\mathbb{E}_{q(z|x)}[\log p(x|z)]$

داده‌های X از روی متغیرهای نهان Z دارد.

• واگرایی KL بین توزیع پسین تقریب $q(z|x)$ و توزیع پیشین $p(z)$ $\text{KL}(q(z|x)||p(z))$

که نقش یک منظم‌کننده را بازی می‌کند.

• β : پارامتری که تعادل بین بازسازی و جداسازی عوامل نهان را کنترل می‌کند.

تجزیه واگرایی KL به دو بخش:

هدف این است که نشان دهیم واگرایی KL در تابع هزینه β -VAE به دو بخش تقسیم می‌شود:

اطلاعات متقابل (Mutual Information) بین X و Z : $I_q(x; z)$

واگرایی KL بین توزیع حاشیه‌ای $q(z)$ و توزیع پیشین $p(z)$ $\text{KL}(q(z)||p(z))$

تعریف توزیع‌های مورد نیاز:

• توزیع داده‌های مشاهده شده: $p_{\text{data}}(x)$

• توزیع پسین تقریب: $q(z|x)$

• توزیع حاشیه‌ای: $q(z)$

$$q(z) = \int q(z|x) p_{\text{data}}(x) dx$$

2. شروع از واگرایی KL در تابع هزینه:

$$\mathbb{E}_{p_{\text{data}}(x)}[\text{KL}(q(z|x)||p(z))] = \int p_{\text{data}}(x) \int q(z|x) \log \frac{q(z|x)}{p(z)} dz dx$$

3. اضافه کردن و کم کردن :

با ضرب و تقسیم $q(z)$ در داخل لگاریتم:

$$\log \frac{q(z|x)}{p(z)} = \log \frac{q(z|x)}{q(z)} + \log \frac{q(z)}{p(z)}$$

4. جدا کردن دو قسمت:

بنابراین، واگرایی KL را می‌توان به صورت زیر نوشت:

$$\mathbb{E}_{p_{\text{data}}(x)}[\text{KL}(q(z|x)||p(z))] = \mathbb{E}_{p_{\text{data}}(x)} \left[\int q(z|x) \log \frac{q(z|x)}{q(z)} dz \right] + \int q(z) \log \frac{q(z)}{p(z)} dz$$

• قسمت اول: اطلاعات متقابل بین X و Z :

$$I_q(x; z) = \mathbb{E}_{p_{\text{data}}(x)} [\text{KL}(q(z|x)||q(z))]$$

- قسمت دوم: واگرایی KL بین $p(z)$ و $q(z)$

$$\text{KL}(q(z)||p(z)) = \int q(z) \log \frac{q(z)}{p(z)} dz$$

: بنابراین

$$\mathbb{E}_{p_{\text{data}}(x)}[\text{KL}(q(z|x)||p(z))] = I_q(x; z) + \text{KL}(q(z)||p(z))$$

در مقاله Disentangling by Factorising، نویسنده‌گان از این تجزیه برای بهبود معاوضه بین کیفیت بازسازی و جداسازی عوامل نهان استفاده کردند.

در β -VAE، افزایش β منجر به افزایش هر دو اصطلاح $\text{KL}(q(z)||p(z))$ و $I_q(x; z)$ می‌شود. این امر باعث کاهش اطلاعات متقابل بین X و Z می‌شود که به نوبه خود می‌تواند کیفیت بازسازی را کاهش دهد.

:FactorVAE راه حل در

نویسنده‌گان FactorVAE پیشنهاد می‌کنند که به جای پنالتی دادن به کل واگرایی KL، فقط بر روی واگرایی KL بین $p(z)$ و $q(z)$ تمرکز کنیم، زیرا این اصطلاح مسئول جداسازی عوامل نهان است.

:FactorVAE تابع هزینه

$$\mathcal{L}_{\text{FactorVAE}} = \mathbb{E}_{q(z|x)}[\log p(x|z)] - \mathbb{E}_{p_{\text{data}}(x)}[\text{KL}(q(z|x)||p(z))] - \gamma \text{TC}(q(z))$$

که در آن:

$q(z)$ ، که معیاری برای همبستگی کل (Total Correlation) توزیع $\text{TC}(q(z))$ •

اندازه‌گیری وابستگی‌های آماری بین متغیرهای نهان است.

• ۷: پارامتری که وزن پنالتی همبستگی کل را تنظیم می‌کند.

همبستگی کل به صورت زیر تعریف می‌شود:

$$\text{TC}(q(z)) = \text{KL}(q(z)||\prod_{j=1}^d q(z_j))$$

این اصطلاح اندازه می‌گیرد که تا چه حد توزیع $q(z)$ از حاصل ضرب توزیع‌های حاشیه‌ای مستقل

$q(z_j)$ فاصله دارد. بنابراین، پنالتی دادن به این باعث می‌شود که توزیع $q(z)$ به یک توزیع مستقل نزدیک شود، که به جداسازی عوامل نهان کمک می‌کند.

با استفاده از تجزیه و آگرایی KL، تابع هزینه FactorVAE به صورت زیر نوشته می‌شود:

$$\mathcal{L}_{\text{FactorVAE}} = \mathbb{E}_{q(z|x)}[\log p(x|z)] - I_q(x; z) - \text{KL}(q(z)||p(z)) - \gamma \text{TC}(q(z))$$

اما از آنجایی که $\text{KL}(q(z)||p(z))$ در واقع بخشی از $\text{TC}(q(z))$ است، می‌توانیم پنالتی را مستقیماً

بر روی $I_q(x; z)$ آسیبی برسانیم. اعمال کنیم بدون اینکه به $\text{TC}(q(z))$ مزیت اصلی:

با این روش، FactorVAE می‌تواند جداسازی عوامل نهان را بهبود بخشد بدون اینکه کیفیت بازسازی را به طور قابل توجهی کاهش دهد، زیرا اطلاعات متقابل بین X و Z حفظ می‌شود.

جمع‌بندی:

- در β -VAE: پنالتی دادن به کل واگرایی KL باعث کاهش اطلاعات متقابل $I(q(x;z) - q(z|x))$ می‌شود که می‌تواند به کیفیت بازسازی آسیب برساند.
- در FactorVAE: با تجزیه واگرایی KL و پنالتی دادن به اصطلاح همبستگی کل $TC(q(z))$ می‌توانیم وابستگی‌های بین متغیرهای نهان را کاهش داده و جداسازی را بهبود دهیم، در حالی که اطلاعات متقابل حفظ می‌شود.
- نتیجه: FactorVAE یک معاوضه بهتر بین کیفیت بازسازی و جداسازی عوامل نهان ارائه می‌دهد.

سؤال دوم تئوری :

بخش سوم

مقدمه:

درتابع هدف FactorVAE، واگرایی KL بین توزیعهای $q(z)$ و $\bar{q}(z)$ که $\bar{q}(z) = \prod_j q(z_j)$ است حضور دارد:

$$\mathcal{L}_{\text{FactorVAE}} = \frac{1}{N} \sum_{i=1}^N \left[\mathbb{E}_{q(z|x^{(i)})} [\log p(x^{(i)}|z)] - \text{KL}(q(z|x^{(i)})||p(z)) - \gamma \text{KL}(q(z)||\bar{q}(z)) \right]$$

محاسبه مستقیم توزیعهای $q(z)$ و $\bar{q}(z)$ نیاز به انتگرال‌گیری‌های پیچیده و محاسبات سنگین دارد، زیرا $q(z|x)$ یک ترکیب (میانگین) از توزیعهای $q(z)$ بر روی تمام داده‌ها است:

$$q(z) = \int q(z|x) p_{\text{data}}(x) dx$$

بنابراین، محاسبه دقیق این توزیعها در عمل غیرممکن است.

روش‌های پیشنهادی برای تقریب $q(z)$ و $\bar{q}(z)$:

نویسندهای مقاله Disentangling by Factorising چندین روش برای تخمین $q(z)$ و $\bar{q}(z)$ پیشنهاد کردند:

1. استفاده از نمونه‌گیری مستقیم از $q(z)$

با انتخاب تصادفی یک نمونه داده $x^{(i)}$ و سپس نمونه‌گیری از $q(z|x^{(i)})$ ، می‌توان

از $q(z)$ نمونه‌برداری کرد.

2. استفاده از تقریب‌های مبتنی بر دسته (Batch) برای $q(z)$:

با استفاده از نمونه‌های موجود در یک دسته (Batch) به صورت

می‌توان تقریب‌هایی برای $q(z)$ به دست آورد.

اما این روش به دلیل نیاز به Batch‌های بزرگ و محاسبات سنگین، عملً کارآمد نیست.

3. استفاده از ترفند جابجایی ابعاد برای تخمین $\bar{q}(z)$:

با جابجا کردن تصادفی مقادیر هر بعد Z بین نمونه‌های یک Batch، می‌توان نمونه‌هایی از

به دست آورد.

این روش باعث می‌شود که وابستگی بین ابعاد از بین برود و توزیع حاصل تقریباً برابر باشد.

$\bar{q}(z)$

4. استفاده از ترفند نسبت چگالی (Density Ratio Trick) و آموزش یک شبکه (Discriminator):

با استفاده از نمونه‌های $\bar{q}(z)$ و $q(z)$ ، می‌توان یک Discriminator را آموزش داد که

بین این دو توزیع تمایز قائل شود.

سپس با استفاده از خروجی Discriminator، می‌توان واگرایی KL را تخمین زد.

روش انتخابی و توضیح آن:

روش انتخابی توسط نویسندها، استفاده از ترفندهای نسبت چگالی و آموزش یک **Discriminator** است.

توضیح روش:

1. نمونه‌گیری از $q(z)$:

- ابتدا یک Batch از داده‌های $x^{(i)}$ را انتخاب می‌کنیم.
 - از هر $x^{(i)}$ نمونه‌ای از Z را با استفاده از $q(z|x^{(i)})$ به دست می‌آوریم.
2. ایجاد نمونه‌های $\bar{q}(z)$:

- در همان Batch، برای هر بعد j مقادیر Z را بین نمونه‌ها به صورت تصادفی جابجا می‌کنیم.
- این عمل باعث می‌شود که همبستگی بین ابعاد از بین برود و نمونه‌های حاصله از

باشند.

3. آموزش **Discriminator**:

- یک شبکه **Discriminator** تعریف می‌کنیم که ورودی آن Z است و خروجی آن احتمال

است که Z از $D(z)$ آمده باشد (در مقابل $\bar{q}(z)$).

- هدف **Discriminator**، تمایز قائل شدن بین نمونه‌های $q(z)$ و $\bar{q}(z)$ است.

4. تخمین واگرایی KL با استفاده از نسبت چگالی:

- از خروجی‌های **Discriminator** برای تخمین نسبت چگالی $\bar{q}(z)$ استفاده می‌کنیم.

طبق رابطه:

$$\text{KL}(q(z) \parallel \bar{q}(z)) = \mathbb{E}_{q(z)} \left[\log \frac{q(z)}{\bar{q}(z)} \right] \approx \mathbb{E}_{q(z)} \left[\log \frac{D(z)}{1 - D(z)} \right]$$

5. بهروزرسانی پارامترها:

- پارامترهای VAE با استفاده از گرادیان تابع هدف بهروز می‌شوند، که شامل اصطلاح تخمینی KL است.

- پارامترهای Discriminator با استفاده از داده‌های $\bar{q}(z)$ و $q(z)$ بهروز می‌شوند تا تمایزدهی بهتری انجام دهند.

مزایای روش انتخاب:

- عدم نیاز به محاسبات سنگین:

- با استفاده از این روش، نیاز به محاسبه مستقیم $\bar{q}(z)$ و $q(z)$ از بین می‌رود.

- کارایی عملی:

- امکان آموزش مدل به صورت عملی و کارآمد فراهم می‌شود.

- حفظ جریان گرادیان:

- این روش به ما اجازه می‌دهد تا گرادیان‌ها را از طریق شبکه VAE به درستی محاسبه کنیم.

جمع‌بندی:

- روش‌های پیشنهاد شده:

- استفاده از نمونه‌گیری مستقیم، تقریب‌های مبتنی بر Batch، جابجایی ابعاد، و استفاده .Discriminator از

- روش انتخابی:**

- استفاده از ترفندهای نسبت چگالی و آموزش یک Discriminator برای تخمین واگرایی KL

بین $\bar{q}(z)$ و $q(z)$.

- توضیح مختصر:**

- با نمونه‌گیری از $\bar{q}(z)$ و ایجاد $q(z)$ از طریق جابجایی ابعاد، یک Discriminator

آموزش می‌دهیم تا نسبت چگالی را تخمین بزند و از آن برای محاسبه واگرایی KL

استفاده منکریم.

مراجع

- Kim, H., & Mnih, A. (2018).** *Disentangling by Factorising*. Proceedings of the 35th International Conference on Machine Learning (ICML).
- Chen, T. Q., et al. (2018).** *Isolating Sources of Disentanglement in Variational Autoencoders*. Advances in Neural Information Processing Systems (NeurIPS).
- Kingma, D. P., & Welling, M. (2014).** *Auto-Encoding Variational Bayes*. Proceedings of the International Conference on Learning Representations (ICLR).
- Higgins, I., Matthey, L., Pal, A., et al. (2017).** *beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework*. International Conference on Learning Representations (ICLR).
- Boyd, S., & Vandenberghe, L. (2004).** *Convex Optimization*. Cambridge University Press.
- Blei, D. M. (Lecture Notes). *Variational Inference*. Sections on KL Divergence and Mean Field Variational Inference.
- Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. Sections on Variational Inference and Gaussian Distributions.

