

دانشگاه تهران  
پردیس دانشکده‌های فنی  
دانشکده مهندسی برق و کامپیوتر

تمرين شماره: ۲  
مدل‌های مولد عمیق

نام و نام خانوادگی: آرتین توسلی

شماره دانشجویی: ۸۱۰۱۰۲۵۴۳

نیمسال اول  
سال تحصیلی ۴۰۴-۴۰۵

# فهرست مطالب

٤	Normalizing Flow ۱
۴	۱.۱ تئوری
۴	۱.۱.۱ تغییر متغیر
۶	۲.۱.۱ محاسبه دترمینان ژاکوبین
۷	۲.۱ پیاده‌سازی عملی
۷	۱.۲.۱ معماری مدل
۹	۲.۲.۱ آموزش مدل و تولید عکس
۱۳	۳.۱ آشکارسازی ناهنجاری
۱۶	CycleGAN ۲
۱۶	۱.۲ تئوری
۱۶	۱.۱.۲تابع زیان مدل
۱۸	۲.۱.۲ معماری و فرآیند آموزش
۲۰	۳.۱.۲ محدودیت‌های مدل
۲۰	۲.۲ پیاده‌سازی عملی
۲۰	۱.۲.۲ آماده‌سازی داده‌ها
۲۰	۲.۲.۲ فرآیند آموزش
۲۱	۲.۲.۲ تحلیل خروجی‌های مدل
۲۲	۴.۲.۲ تحلیل نمودارهای زیان
۲۲	۵.۲.۲ بخش امتیازی

## فهرست تصاویر

- |              |   |     |
|--------------|---|-----|
| ۱۰ . . . . . | نمونه تصاویر واقعی از دیتاست MVTecAD (Capsule)                                      | ۱.۱ |
| ۱۱ . . . . . | نمودار همگرایی تابع هزینه Training و Validation در طول آموزش                        | ۲.۱ |
| ۱۱ . . . . . | تصاویر تولید شده توسط مدل MAF پس از آموزش   | ۳.۱ |
| ۱۴ . . . . . | توزع امتیازات ناهنجاری برای داده‌های سالم و ناهنجار. جدایی دو توزع قابل مشاهده است. | ۴.۱ |
| ۱۵ . . . . . | منحنی ROC برای تشخیص ناهنجاری. سطح زیر منحنی برابر با ۰.۶۷ است.                     | ۵.۱ |
| ۲۴ . . . . . | ایپاک ۱ برای دیتاست Apple2Orange  | ۱.۲ |
| ۲۵ . . . . . | ایپاک ۱۰ برای دیتاست Apple2Orange   | ۲.۲ |
| ۲۶ . . . . . | ایپاک ۲۰ برای دیتاست Apple2Orange   | ۳.۲ |
| ۲۷ . . . . . | نمودارهای زیان برای دیتاست Apple2Orange   | ۴.۲ |
| ۲۸ . . . . . | نمونه تصاویر دیتاست Summer2Winter   | ۵.۲ |
| ۲۹ . . . . . | ایپاک ۱ برای دیتاست Summer2Winter   | ۶.۲ |
| ۳۰ . . . . . | ایپاک ۱۰ برای دیتاست Summer2Winter  | ۷.۲ |
| ۳۱ . . . . . | ایپاک ۲۰ برای دیتاست Summer2Winter  | ۸.۲ |
| ۳۲ . . . . . | نمودارهای زیان برای دیتاست Summer2Winter  | ۹.۲ |

## فهرست جداول

- ۱۰ ..... هایپرپارامترهای آموزش مدل MAF
- ۱۱ ..... هایپرپارامترهای استفاده شده در آموزش مدل CycleGAN

# سوال ۱

## Normalizing Flow

### ۱.۱ تئوری

#### ۱.۱.۱ تغییر متغیر

در این سوال از فرمول تغییر متغیر تک بعدی زیر استفاده شده است.

$$P_X(x) = P_Z(f^{-1}(x)) \times \left| \frac{df^{-1}(x)}{dx} \right| \quad (1.1)$$

که در آن  $f^{-1}(x)$  تابع معکوس تبدیل است.

#### بخش الف

۱. تئورى

۱. تابع چگالى  $P_Z(z)$

$$P_Z(z) = \begin{cases} 1 & \text{for } 0 \leq z \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

۲. تابع تبدیل معکوس  $f^{-1}(x)$

$$Z = \frac{x-1}{2}$$

۳. قدر مطلق مشتق

$$\left| \frac{dZ}{dx} \right| = \left| \frac{d}{dx} \left( \frac{x-1}{2} \right) \right| = \left| \frac{1}{2} \right| = \frac{1}{2}$$

۴. دامنه متغیر  $X$

$$0 \leq Z \leq 1$$

$$0 \leq \frac{x-1}{2} \leq 1$$

$$\Rightarrow 0 \leq x-1 \leq 2$$

$$\Rightarrow 1 \leq x \leq 3$$

۵. تابع چگالى احتمال  $P_X(x)$

$$P_X(x) = P_Z \left( \frac{x-1}{2} \right) \times \frac{1}{2}$$

$$P_X(x) = 1 \times \frac{1}{2} = \frac{1}{2} \quad (\text{for } 1 \leq x \leq 3)$$

$$P_X(x) = \begin{cases} \frac{1}{2} & \text{for } 1 \leq x \leq 3 \\ 0 & \text{otherwise} \end{cases}$$

بخش ب

۱. تابع چگالى  $P_Z(z)$

$$P_Z(z) = \begin{cases} \frac{1}{2} & \text{for } 0 \leq z \leq 2 \\ 0 & \text{otherwise} \end{cases}$$

٢. تابع تبديل معكوس  $f^{-1}(x)$

$$Z = \ln(x)$$

٣. قدر مطلق مشتق

$$\left| \frac{dZ}{dx} \right| = \left| \frac{d}{dx} (\ln(x)) \right| = \left| \frac{1}{x} \right| = \frac{1}{x}$$

٤. دامنه متغير  $X$ :

$$0 \leq Z \leq 2$$

$$e^0 \leq e^Z \leq e^2$$

$$\Rightarrow 1 \leq X \leq e^2$$

٥. تابع چگالي احتمال  $P_X(x)$

$$P_X(x) = P_Z(\ln(x)) \times \frac{1}{x}$$

$$P_X(x) = \frac{1}{2} \times \frac{1}{x} = \frac{1}{2x} \quad (\text{for } 1 \leq x \leq e^2)$$

$$P_X(x) = \begin{cases} \frac{1}{2x} & \text{for } 1 \leq x \leq e^2 \\ 0 & \text{otherwise} \end{cases}$$

## ٢.١.١ محاسبه دترمینان ژاكوبين

### بخش الف

$$x_1 = z_1$$

$$x_2 = z_2 - m(x_1) = z_2 - m(z_1)$$

ماترييس ژاكوبين  $J = \frac{\partial \mathbf{x}}{\partial \mathbf{z}}$  به صورت زير است:

$$J = \begin{pmatrix} \frac{\partial x_1}{\partial z_1} & \frac{\partial x_1}{\partial z_2} \\ \frac{\partial x_2}{\partial z_1} & \frac{\partial x_2}{\partial z_2} \end{pmatrix} = \begin{pmatrix} I & 0 \\ -\frac{\partial m(z_1)}{\partial z_1} & I \end{pmatrix}$$

## ۱.۲. پیاده‌سازی عملی

این ماتریس پایین-مثلثی است. دترمینان آن برابر با حاصلضرب دترمینان بلوک‌های قطری است:

$$\det(J) = \det(I) \times \det(I) = 1$$

### بخش ب

$$x_1 = z_1$$

$$x_2 = \frac{z_2 - t(x_1)}{s(x_1)} = \frac{z_2 - t(z_1)}{s(z_1)}$$

ماتریس ژاکوبین  $J = \frac{\partial \mathbf{x}}{\partial \mathbf{z}}$  به صورت زیر است:

$$J = \begin{pmatrix} \frac{\partial x_1}{\partial z_1} & \frac{\partial x_1}{\partial z_2} \\ \frac{\partial x_2}{\partial z_1} & \frac{\partial x_2}{\partial z_2} \end{pmatrix} = \begin{pmatrix} I & 0 \\ \frac{\partial x_2}{\partial z_1} & \frac{1}{s(z_1)}I \end{pmatrix}$$

با فرض اینکه  $x_2$  و  $z_2$  بردارهایی به طول  $d$  باشند، دترمینان ماتریس پایین-مثلثی برابر است با:

$$\det(J) = \det(I) \times \det \left( \frac{1}{s(z_1)} I \right) = 1 \times \left( \frac{1}{s(z_1)} \right)^d = \left( \frac{1}{s(z_1)} \right)^d$$

## ۲.۱ پیاده‌سازی عملی

### ۱.۲.۱ معما ری مدل

برای پیاده‌سازی مدل Masked Autoregressive Flow (MAF)، ابتدا بلوک اصلی سازنده‌ی آن یعنی MADE پیاده‌سازی شد. هدف از این مدل، تخمین چگالی توأم ( $\mathbf{x}$ )  $p$  با تجزیه آن به احتمالات شرطی تک‌بعدی  $p(x_i|x_{1:i-1})$  است. برای تضمین خاصیت autoregressive در یک شبکه عصی تمام‌متصل، از تکنیک Masking روی وزن‌های شبکه استفاده شده است. جزئیات پیاده‌سازی لایه Masked Linear به جای استفاده از لایه‌های خطی معمولی، کلاسی تعریف شد که یک ماتریس ماسک همان‌دازه با ماتریس وزن‌ها نگه می‌دارد. در هنگام انتشار رو به جلو (Forward Pass)، وزن‌ها در ماسک ضرب می‌شوند:

$$\mathbf{y} = (\mathbf{W} \odot \mathbf{M})\mathbf{x} + \mathbf{b}$$

## ۱. ۲. پیاده‌سازی عملی

۸

که در آن  $\odot$  ضرب درایه‌ای است. ماسک‌ها به گونه‌ای مقداردهی می‌شوند که نورون خروجی  $k$  تنها به ورودی‌های با اندازه کمتر از  $k$  دسترسی داشته باشد.

**ساختار شبکه:** مطابق با معماری پیشنهادی در صورت تمرین، شبکه MADE شامل سه لایه به شرح زیر است:

۱. لایه ورودی به مخفی ۱: تعداد ورودی برابر با ابعاد تصویر مسطح شده ( $3 \times 128 \times 128$ ) و خروجی ۵۱۲ نورون.

۲. لایه مخفی ۱ به مخفی ۲: ورودی ۵۱۲ و خروجی ۵۱۲ نورون.

۳. لایه مخفی ۲ به خروجی: ورودی ۵۱۲ و خروجی برابر با دو برابر ابعاد ورودی (برای تولید پارامترهای میانگین  $\mu$  و لگاریتم انحراف معیار  $\alpha$ ).

برای تضمین اینکه شبکه در هنگام پیش‌بینی بعد  $k$ -ام ( $x_k$ )، هیچ اطلاعی از خود  $x_k$  یا ابعاد آینده نداشته باشد، از مکانیزم زیر استفاده شده است:

• به هر نورون یک عدد به نام «درجه» (degree) اختصاص می‌یابد که نشان می‌دهد آن نورون مجاز است حداکثر تا چه بُعدی از ورودی را مشاهده کند.

• در لایه‌های مخفی: اتصالات به گونه‌ای ماسک می‌شوند که نورون‌ها فقط از نورون‌های لایه قبل با درجه‌ی کمتر یا مساوی ورودی بگیرند (انتقال اطلاعات مجاز).

$$m_k^l \geq m_d^{l-1}$$

• در لایه خروجی: شرط سخت‌گیرانه‌تر است؛ نورون خروجی مربوط به بُعد  $k$ ، تنها اجازه دارد به نورون‌هایی وصل شود که درجه آن‌ها اکیداً کمتر از  $k$  باشد. این تضمین می‌کند که برای ساخت  $x_k$  اطلاعات در اختیار شبکه قرار نمی‌گیرد.

$$m_k^{out} > m_d^{last\_hidden}$$

مدل MAF از ترکیب چندین تبدیل MADE به صورت متواالی تشکیل شده است تا انعطاف‌پذیری مدل در تخمین توزیع‌های پیچیده افزایش یابد. طبق خواسته مسأله، از ۷ بلوک تبدیل استفاده شده است.

## ۱.۲. پیاده‌سازی عملی

۹

فرآیند عبور رو به جلو (تخمین چگالی): در فاز آموزش، هدف محاسبه  $P(X)$  است. مدل MAF تبدیل  $Z \rightarrow X : f$  را مدل می‌کند که در آن  $Z$  دارای توزیع نرمال استاندارد است. با فرض اینکه  $x$  ورودی یک لایه و  $u$  خروجی آن باشد:

$$u_i = (x_i - \mu_i(x_{<i})) \cdot \exp(-\alpha_i(x_{<i})) \quad (2.1)$$

که در آن  $\mu$  و  $\alpha$  خروجی‌های شبکه MADE هستند. از آنجا که این پارامترها تنها به مقادیر پیشین ورودی وابسته هستند، ماتریس ژاکوبین این تبدیل مثلثی پایین است و دترمینان آن برابر با حاصل ضرب درایه‌های روی قطر اصلی است:

$$\det \left| \frac{\partial \mathbf{u}}{\partial \mathbf{x}} \right| = \exp \left( - \sum_i \alpha_i \right) \quad (3.1)$$

جایگشت ابعاد (Permutation): از آنجا که در یک لایه MADE، پیکسل‌های انتهایی تاثیری بر پیکسل‌های ابتدایی ندارند، اگر ترتیب ورودی‌ها در تمام ۷ لایه ثابت باشد، مدل‌سازی ناقص خواهد بود. برای رفع این مشکل، بین هر دو بلوك متوالی، ترتیب ابعاد بردار ورودی معکوس (Flip) می‌شود. این کار باعث می‌شود تمام ابعاد تصویر در لایه‌های مختلف بر یکدیگر اثر بگذارند و یک توزیع توأم کامل شکل گیرد.

نهایتاً تابع هدف برای آموزش مدل، بیشینه‌سازی Log-Likelihood داده‌های آموزشی است:

$$\log P_X(x) = \log P_Z(f(x)) + \sum_{k=1}^7 \log |\det J_k| \quad (4.1)$$

## ۲.۲.۱ آموزش مدل و تولید عکس

در این بخش، مدل MAF برای تخمین چگالی توزیع داده‌های مجموعه Capsule (کلاس MV TecAD) پیاده‌سازی و آموزش داده شده است. قبل از آموزش، نمونه‌هایی از داده‌های آموزشی برای اطمینان از صحت بارگذاری دیتابست در شکل ۱.۱ نمایش داده شده است.

## ۱.۲. پیاده‌سازی عملی

۱۰



شکل ۱.۱: نمونه تصاویر واقعی از دیتاست .MVTecAD (Capsule)

جدول هایپرپارامتر های استفاده شده نیز در جدول ۱.۱ نیز آورده شده است.

جدول ۱.۱: هایپرپارامترهای آموزش مدل MAF

پارامتر (Parameter)	مقدار (Value)	توضیحات (Value)
Input Dimension	$128 \times 128 \times 3$	ابعاد تصاویر ورودی
Hidden Dimensions	$[512, 512, 512]$	تعداد نورون های لایه های مخفی در هر بلوک MADE
Number of Blocks	7	تعداد بلوک های MAF پشت سر هم
Batch Size	64	اندازه دسته
Learning Rate	$1 \times 10^{-4}$	نرخ یادگیری بهینه ساز Adam
Epochs	50	تعداد دوره های آموزش
Validation Split	0.1	درصد داده های اعتبارسنجی

نمودار تغییرات تابع هزینه (Loss) در طول ۵۰ اپاک آموزش در شکل ۲.۱ آورده شده است.

- همان‌طور که مشاهده می‌شود، مدل با سرعت بسیار زیادی در ایپاک‌های اولیه همگرا شده و مقدار

به شدت کاهش می‌یابد.

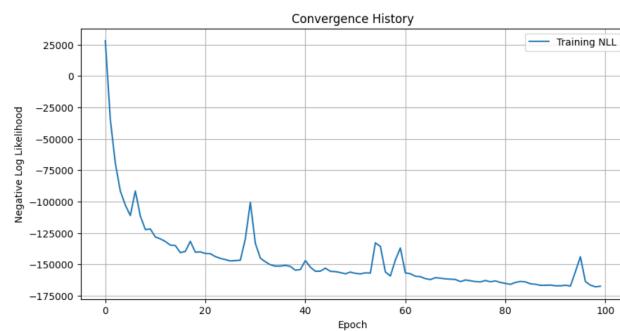
- مقادیر منفی بزرگ برای Loss در مدل های مبنابر likelihood مشکلی ندارد. (زیرا Log-Likelihood روی چگالی احتمال محاسبه می‌شود و می‌تواند مثبت باشد، پس منفی آن منفی می‌شود.)

پس از پایان آموزش، با نمونه برداری از توزیع نرمال ( $I \sim \mathcal{N}(0, Z)$  و عبور دادن آن از شبکه معکوس  $(x = f^{-1}(z))$ ، تصاویر جدیدی تولید شد که در شکل ۳.۱ قابل مشاهده است.

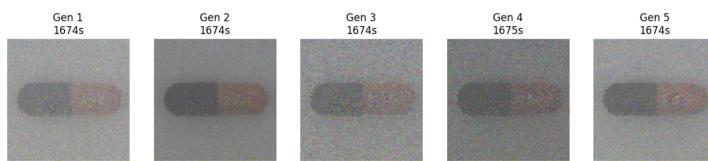
- مدل موفق شده است ساختار کلی کپسول‌ها (دو نیمه رنگی و سفید) و جهت‌گیری آن‌ها را یاد بگیرد.

- بافت پس‌زمینه تا حدودی بازسازی شده است، هرچند تصاویر نسبت به نمونه‌های واقعی دارای نویز هستند و شفافیت کامل را ندارند.

- زمان آموزش: ۱۷۸۲ ثانیه (تقریباً ۳۰ دقیقه) برای ۵۰ اپاک آموزش طول کشید. بسیار سریع



شکل ۲.۱: نمودار همگرایی تابع هزینه (Validation و Training) در طول آموزش.



شکل ۳.۱: تصاویر تولید شده توسط مدل MAF پس از آموزش.

است. دلیل این امر آن است که در معماری MAF، فرآیند محاسبه  $f(x) = z$  (که برای محاسبه Likelihood در آموزش استفاده می‌شود) قابل موازی‌سازی است. ما تمام پیکسل‌های تصویر ورودی  $x$  را داریم و می‌توانیم تمامی شرط‌ها را همزمان محاسبه کنیم.

- زمان تولید: (Sampling) به طور میانگین ۱۶۷۵ ثانیه (قریباً ۲۸ دقیقه) برای تولید کردن هر عکس طول کشید. بسیار کند است. برای تولید ۵ تصویر، زمان قابل توجهی صرف شد. دلیل این کندی خاصیت Autoregressive مدل است.

تولید عکس در این مدل بسیار زمان بر است چون فرآیند تولید به صورت ترتیبی (Sequential) است. برای تولید پیکسل  $x_i$ , مدل باید تمام پیکسل‌های قبلی  $x_{i-1}, x_{i-2}, \dots, x_1$  را تولید کرده باشد.

$$p(x) = \prod_{i=1}^D p(x_i | x_{<i})$$

بنابراین برای یک تصویر با ابعاد  $3 \times 128 \times 128$ , مدل باید ۱۵۲ بار به صورت پشت سر هم اجرا شود تا یک تصویر کامل ساخته شود. امکان موازی‌سازی در زمان تولید وجود ندارد.

تفاوت چشمگیر در سرعت تولید (Sampling) بین دو مدل MAF و IAF ناشی از ساختار وابستگی‌های

## ۱.۲. پیاده‌سازی عملی

۱۲

شرطی در فرمول بازسازی داده‌هاست.

- ۱. مدل MAF (تولید ترتیبی و کند): در MAF، تبدیل از فضای نهان  $z$  به فضای داده  $x$  (فرآیند تولید) از رابطه بازگشتی زیر پیروی می‌کند:

$$x_i = z_i \cdot \sigma_i(x_1, x_2, \dots, x_{i-1}) + \mu_i(x_1, x_2, \dots, x_{i-1}) \quad (5.1)$$

همان‌طور که در معادله بالا مشخص است، برای محاسبه  $x_i$ ، شبکه عصبی (که  $\mu$  و  $\sigma$  را تخمین می‌زنند) نیاز دارد که مقادیر  $x_1$  تا  $x_{i-1}$  را به عنوان ورودی دریافت کند.

- این یعنی ما نمی‌توانیم  $x_i$  را محاسبه کنیم مگر اینکه محاسبه  $x_{i-1}$  تمام شده باشد.
- بنابراین فرآیند تولید ذاتاً سری است و قابلیت موازی‌سازی ندارد. برای یک تصویر با ابعاد  $D$ ، باید بار شبکه عصبی اجرا شود.  $(O(D))$

- ۲. مدل IAF (تولید موازی و سریع): مدل Inverse Autoregressive Flow با معکوس کردن این وابستگی‌ها طراحی شده است. در IAF، پارامترهای توزیع  $x_i$  به جای اینکه به مقادیر قبلی  $x$  وابسته باشند، به مقادیر قبلی  $z$  وابسته‌اند:

$$x_i = z_i \cdot \sigma_i(z_1, z_2, \dots, z_{i-1}) + \mu_i(z_1, z_2, \dots, z_{i-1}) \quad (6.1)$$

در زمان تولید، ما ابتدا کل بردار نویز  $z$  را از توزیع نرمال نمونه‌برداری می‌کنیم  $(z \sim \mathcal{N}(0, I))$ .

- چون تمامی مقادیر  $z$  (از  $z_1$  تا  $z_D$ ) از همان لحظه شروع در دسترس هستند، شبکه عصبی می‌تواند  $(z)$  و  $(\mu(z))$  را برای همه ابعاد  $n$  به صورت همزمان محاسبه کند.
- هیچ وابستگی بازگشتی به خروجی‌های محاسبه‌شده وجود ندارد.
- بنابراین فرآیند تولید کاملاً موازی است و کل تصویر تنها با یک بار اجرای شبکه تولید می‌شود  $(O(1))$ .

مدل MAF در محاسبه چگالی احتمال  $(z \rightarrow x)$  سریع است (چون  $x$  معلوم است و موازی می‌شود) و برای آموزش مناسب است و در anomaly detection کاربرد دارد. در مقابل، IAF در فرآیند تولید Real-time Generation  $(z \rightarrow x)$  سریع است (چون  $z$  معلوم است و موازی می‌شود) و برای کاربردهای مناسب‌تر است.

## ۳.۱ آشکارسازی ناهنجاری

۱. چرا از معیار دقت (Accuracy) استفاده نمی‌شود؟ در مسائل تشخیص ناهنجاری، توزیع داده‌ها عموماً بسیار (Imbalanced) است. برای مثال، ممکن است ۹۹٪ داده‌ها سالم و تنها ۱٪ ناهنجار باشند. در چنین شرایطی، اگر مدلی تمام داده‌ها را «سالم» پیش‌بینی کند، به دقت ۹۹٪ می‌رسد اما عملأ هیچ ناهنجاری‌ای را تشخیص نداده است. معیار جایگزین: از معیار سطح زیر منحنی مشخصه عملکرد سیستم (AUROC) استفاده می‌شود. این معیار مستقل از آستانه‌گذاری (Threshold) است و توانایی مدل در تفکیک دو کلاس را در تمام نقاط کاری نشان می‌دهد.

۲. مفهوم Anomaly Score چیست؟ امتیاز ناهنجاری، یک مقدار عددی اسکالر است که به هر نمونه داده نسبت داده می‌شود و میزان «غیرعادی» بودن آن را نشان می‌دهد. هرچه این عدد بزرگتر باشد، احتمال اینکه داده ناهنجار باشد بیشتر است. در مدل‌های بنابر likelihood، عموماً از منفی احتمال لگاریتمی  $(-\log p(x))$  به عنوان این امتیاز استفاده می‌شود.

۳. استفاده از Normalizing Flow برای تشخیص ناهنجاری: مدل‌های NF روی داده‌های نرمال آموزش می‌بینند تا چگالی احتمال  $p_{\text{normal}}(x)$  را بیشینه کنند. پس از آموزش:

- اگر یک داده سالم به مدل داده شود، مقدار  $p(x)$  بالا خواهد بود (چون مدل آن را دیده است).
- اگر یک داده ناهنجار به مدل داده شود، چون توزیع آن با داده‌های آموزشی متفاوت است، مدل مقدار  $p(x)$  بسیار پایینی برای آن تخمین می‌زند.

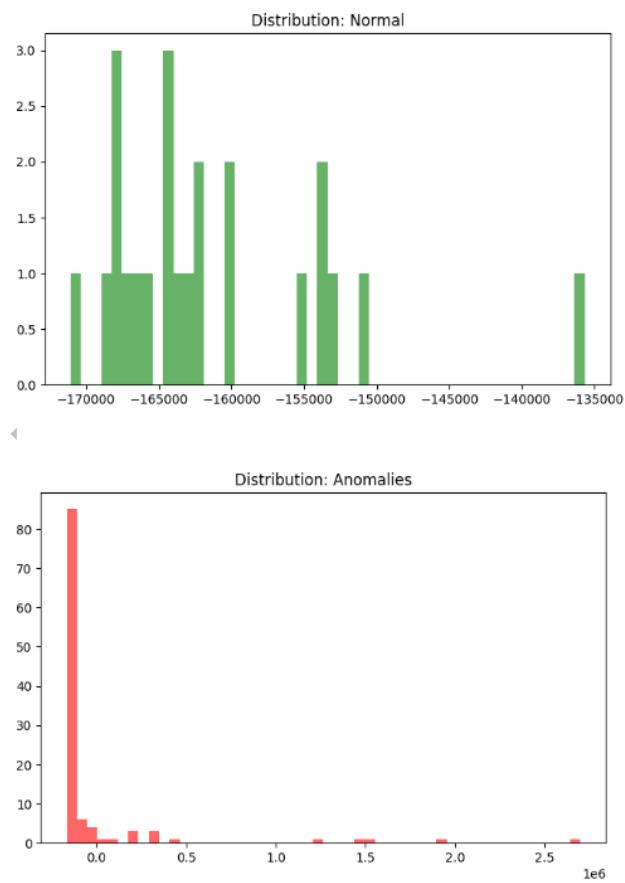
بنابراین با تعریف امتیاز ناهنجاری به صورت  $\mathcal{S} = -\log p_{\theta}(x)$ ، داده‌های سالم امتیاز پایین و ناهنجاری‌ها امتیاز بالا می‌گیرند. برای ارزیابی مدل، داده‌های تست (شامل نمونه‌های سالم و ناهنجار) به مدل داده شد و مقدار NLL برای هر تصویر محاسبه گردید.

توزیع امتیازات ناهنجاری شکل ۴.۱ هیستوگرام امتیازات ناهنجاری را برای دو کلاس سالم (Good) و ناهنجار (Anomaly) نشان می‌دهد.

- همان‌طور که مشاهده می‌شود، میانگین امتیازات داده‌های نرمال (سیز) کمتر از داده‌های ناهنجار (قرمز) است. این نشان می‌دهد مدل توانسته است تفاوت توزیعی بین کپسول‌های سالم و معیوب را درک کند.
- با این حال، همپوشانی قابل توجهی بین دو نمودار وجود دارد که نشان می‌دهد تشخیص برخی ناهنجاری‌ها برای مدل دشوار بوده است.

### ۱۳. آشکارسازی ناهنجاری

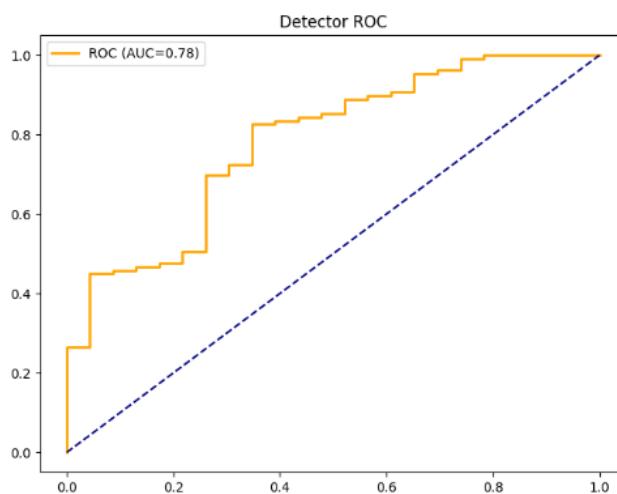
۱۴



شکل ۱۴.۱: توزیع امتیازات ناهنجاری برای داده‌های سالم و ناهنجار. جدایی دو توزیع قابل مشاهده است.

### ROC منحنی و معیار AUC

شکل ۱۵.۱ عملکرد مدل را با معیار AUROC نشان می‌دهد. مدل به عدد  $AUC = 0.78$  رسیده است. این عدد بیانگر این است که اگر به صورت تصادفی یک نمونه سالم و یک نمونه ناهنجار انتخاب کنیم، مدل با احتمال ۷۸٪ امتیاز بالاتری به نمونه ناهنجار می‌دهد.



شکل ۱۵: منحنی ROC برای تشخیص ناهنجاری. سطح زیر منحنی برابر با ۰.۶۷ است.

#### استفاده خطای بازسازی به عنوان معیار تشخیص ناهنجاری

- در VAE: این مدل‌ها (Lossy Bottleneck) هستند و از یک عبور می‌کنند. وقتی مدل روی داده‌های سالم آموزش ببیند، نمی‌تواند جزئیات ناهنجار (مثلاً شکستگی کپسول) را کدگذاری کند، بنابراین هنگام بازسازی، آن جزئیات حذف شده و خطای بازسازی  $(\|\hat{x} - \hat{x}\|^2)$  زیاد می‌شود.
- در Normalizing Flow: این مدل‌ها معکوس‌پذیر و lossless هستند. نگاشت  $f : X \rightarrow Z$  یک تناظر یک‌به‌یک است (Bijective). این یعنی مدل می‌تواند هر داده‌ای (حتی نویز خالص یا ناهنجاری شدید) را به فضای نهان برد و دقیقاً همان را بدون هیچ خطایی بازسازی کند  $(x = f^{-1}(f(x)))$ .

بنابراین در NF، خطای بازسازی همواره نزدیک به صفر است و نمی‌تواند معیاری برای تشخیص ناهنجاری باشد. در عوض، ما از Likelihood استفاده می‌کنیم.

## سوال ۲

# CycleGAN

### ۱.۲ تئوری

#### ۱.۱.۲تابع زیان مدل

۱. چالش داده‌های زوج در یادگیری ناظارتی روش‌های مرسوم یادگیری ناظارتی برای تبدیل تصویر به تصویر (مانند pix2pix) نیازمند جفت داده هستند. یعنی برای هر تصویر در مبدأ، تصویری متناظر در دامنه مقصد وجود داشته باشد. جمع‌آوری چنین داده‌هایی در دنیای واقعی بسیار سخت می‌باشد برای مثال، در تبدیل تصویر سبب به پرتقال، نمی‌توانیم دقیقاً همان سبب را در همان زاویه و نورپردازی به پرتقال تبدیل کنیم تا عکس بگیریم. این چالش استفاده از روش‌های ناظارتی را محدود می‌کند.

۲. قید سازگاری چرخه‌ای (Cycle Consistency) ایده اصلی CycleGAN برای حل مشکل نبود داده‌های زوج، استفاده از دو نگاشت معکوس  $Y \rightarrow X$  و  $G : X \rightarrow Y$  است. قید سازگاری چرخه‌ای بیان می‌کند که اگر یک تصویر را از دامنه  $X$  به  $Y$  ببریم و سپس نتیجه را با نگاشت معکوس به  $X$  برگردانیم، باید به تصویر اولیه برسیم.

$$x \rightarrow G(x) \rightarrow F(G(x)) \approx x \quad (1.2)$$

این قید به صورت ریاضی با نرم  $L1$  بیان می‌شود:

$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)} [\|G(F(y)) - y\|_1] \quad (2.2)$$

۳. اهمیت قید سازگاری چرخه‌ای بدون این قید، شبکه می‌تواند هر تصویری از دامنه  $X$  را به هر تصویر تصادفی و بی‌ارتباط اما واقعی در دامنه  $Y$  نگاشت کند تا descriminator Mode (فریب دهنده) را فریب دهد (Collapse). قید سازگاری چرخه‌ای تضمین می‌کند که محتوای تصویر (شکل، پس‌زمینه، زاویه) حفظ شده و فقط دامنه تغییر کند. سناریوی Mode Collapse: مولد ممکن است متوجه شود که یک تصویر خاص از دامنه  $Y$  (مثلًا تصویر یک سیب قرمز خاص) وجود دارد که تمیزدهنده را به شدت فریب می‌دهد. در این حالت، مولد تصمیم می‌گیرد تمام تصاویر ورودی (چه سیب سبز، چه زرد، چه کوچک، چه بزرگ) را به همان یک تصویر خاص نگاشت کند. چون آن تصویر خروجی بسیار واقعی است، تمیزدهنده راضی است. اما تنوع ورودی‌ها از بین رفته و همه به یک خروجی تبدیل شده‌اند. این همان Mode Collapse است. قید سازگاری چرخه‌ای چگونه جلوی این کار را می‌گیرد؟ قید سازگاری چرخه‌ای ( $\rightarrow G(x) \approx x$ ) مدل را مجبور می‌کند که عملیاتش برگشت‌پذیر باشد. اگر مدل دچار Mode Collapse شود و ۱۰ تصویر مختلف ورودی را به ۱ تصویر یکسان خروجی تبدیل کند، هنگام بازگشت (-Back-translation)، مدل دوم ( $F$ ) نمی‌تواند تشخیص دهد که این ۱ تصویر باید به کدامیک از آن ۱۰ تصویر اولیه برگرد. در نتیجه خطای چرخه‌ای ( $L_{\text{cyc}}$ ) بسیار زیاد خواهد شد.

۴. تابع زیان کامل تابع زیان کامل شامل زیان تخاصمی (برای واقعی به نظر رسیدن تصاویر) و زیان چرخه‌ای است:

$$\begin{aligned} \mathcal{L}(G, F, D_X, D_Y) &= \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) \\ &\quad + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \\ &\quad + \lambda \mathcal{L}_{\text{cyc}}(G, F) \end{aligned} \quad (3.2)$$

ضریب  $\lambda$  اهمیت نسبی حفظ محتوا را تعیین می‌کند. اگر این مقدار بیش از حد بزرگ باشد، مدل آزادی عمل خود را از دست داده و نمی‌تواند تغییرات لازم (مثل تغییر رنگ سیب به پرتقال) را انجام دهد و تصویر خروجی بیش از حد شبیه ورودی می‌شود.

۵. زیان همانی (Identity Loss) زیان همانی بیان می‌کند که اگر یک تصویر واقعی از دامنه مقصد (مثلًا تصویر یک زمستان واقعی) به مولدی داده شود که وظیفه‌اش تولید تصاویر زمستانی است، خروجی باید

همان تصویر ورودی باشد. (مثلاً سیب را به همون سیب تبدیل نمایند)

$$\mathcal{L}_{\text{identity}}(G, F) = \mathbb{E}_{y \sim p_{\text{data}}(y)} [\|G(y) - y\|_1] + \mathbb{E}_{x \sim p_{\text{data}}(x)} [\|F(x) - x\|_1] \quad (4.2)$$

این زیان از تغییرات ناخواسته رنگ و ترکیب‌بندی جلوگیری می‌کند. مثلاً در تبدیل تابستان به زمستان، اگر عکسی خودش برفی است، مدل نباید رنگ آسمان آن را تغییر دهد. این زیان جلوگیری می‌کند از نگاشت‌های ساده و سراسری مولد برای فریب دادن descriminator به عنوان مثال، در تبدیل تصاویر زمستان به تابستان، مولد ممکن است یاد بگیرد که صرفاً با اعمال یک فیلتر رنگی سبز روی کل تصویر (شامل آسمان، کوه و زمین)، توزیع رنگی دامنه مقصد را شبیه‌سازی کند. این امر منجر به تولید تصاویر غیرطبیعی می‌شود که در آن‌ها آسمان یا اجسام پس‌زمینه دچار تغییر رنگ نامطلوب شده‌اند. زیان همانی با اعمال قید  $y \approx G(y)$  (که در آن  $y$  نمونه‌ای واقعی از دامنه مقصد است)، به عنوان یک مکانیزم تنظیم‌کننده عمل می‌کند. منطق این زیان بدین صورت است: ”اگر یک تصویر واقعی تابستانی (دارای آسمان آبی و پوشش گیاهی سبز) به مولد داده شود، خروجی باید دقیقاً همان تصویر ورودی باشد.“

## ۲.۱.۲ معماری و فرآیند آموزش

### ۱. دلیل استفاده از دو مولد مجزا

- دلیل تئوری (برگشت‌پذیری): هدف اصلی CycleGAN یادگیری یک نگاشت یک‌به‌یک بین دو دامنه است. برای اینکه بتوانیم قید سازگاری چرخه‌ای ( $x \rightarrow y \rightarrow x$ ) را اعمال کنیم، نیاز به یک مسیر برگشت داریم. مولد  $G$  وظیفه نگاشت  $Y \rightarrow X$  را دارد، اما بدون مولد دوم  $F$  که وظیفه نگاشت معکوس  $X \rightarrow Y$  را انجام دهد، راهی برای بازگرداندن تصویر و محاسبه خطای بازسازی نداریم. بنابراین وجود دو مولد برای تعریف ریاضیتابع زیان چرخه‌ای ضروری است.
- دلیل عملی (استقلال دامنه‌ها): در بسیاری از کاربردها، تبدیل یک‌طرفه نیست. مثلاً هم می‌خواهیم ”عکس را به نقاشی“ تبدیل کنیم و هم ”نقاشی را به عکس“. داشتن دو مولد مجزا به ما این امکان را می‌دهد که همزمان دو مدل برای هر دو جهت یاد بگیریم که هر کدام متخصص تبدیل در یک جهت خاص هستند.

### ۲. مقایسه معماری مولد با (ResNet vs U-Net) pix2pix

- مدل pix2pix (معماری U-Net): چون pix2pix با داده‌های زوج (Paired) آموزش می‌یابند، ورودی و خروجی از نظر مکانی کاملاً منطبق هستند. معماری U-Net با داشتن Skip Connections

اطلاعات مکانی سطح پایین را مستقیماً از ورودی به خروجی منتقل می‌کند که برای این نوع مسائل عالی است.

- **مدل CycleGAN (معماری ResNet):** در اینجا داده‌ها غیرزوج هستند و ممکن است تغییرات ساختاری و هندسی لازم باشد (مثلًا تغییر فرم بدن اسب). Skip Connections در U-Net ممکن است ساختار ورودی را بیش از حد حفظ کنند و مانع تغییرات لازم شوند. معماری ResNet با یادگیری Residuals و داشتن یک Bottleneck، به شبکه اجازه می‌دهد تا ویژگی‌های معنایی عمیق‌تری را استخراج کرده و تغییرات استایلی پیچیده‌تری را بدون وابستگی شدید به تطابق پیکسل‌به‌پیکسل انجام دهد.

### ۳. هدف استفاده از تمیزدهنده PatchGAN

تمرکز بر جزئیات بالا. تمیزدهنده‌های معمولی اغلب روی ساختار کلی تصویر تمرکز می‌کنند و ممکن است تصاویری که بافت تار دارند را واقعی تشخیص دهند. PatchGAN تصویر را به تکه‌های کوچک (مثلًا  $70 \times 70$ ) تقسیم کرده و هر تکه را جداگانه قضاوت می‌کند. خروجی این شبکه یک ماتریس دو بعدی از اعداد بین ۰ و ۱ است. هر درایه در این ماتریس نشان می‌دهد که پچ متناظر در تصویر ورودی چقدر واقعی به نظر می‌رسد. میانگین این ماتریس به عنوان امتیاز نهایی در نظر گرفته می‌شود. این کار باعث می‌شود مولد مجبور شود بافت‌های ریز و دقیق (مثل چمن، پوست گورخر) را با کیفیت بالا تولید کند.

### ۴. بافر تاریخچه تصاویر (Image History Buffer)

اگر تمیزدهنده فقط با آخرین تصاویر تولید شده توسط مولد آموزش ببیند، ممکن است مولد وارد چرخه نوسان شود. یعنی مولد یک نوع تصویر خاص تولید می‌کند، تمیزدهنده یاد می‌گیرد آن را تشخیص دهد، مولد استراتژی خود را کاملاً عوض می‌کند و تمیزدهنده استراتژی قبلی را فراموش می‌کند (Catas-trophic Forgetting). با ذیجه ۵۰ تصویر تولید شده اخیر در یک بافر و نمونه‌برداری تصادفی از آن‌ها برای آموزش تمیزدهنده، ما "توزيع" تصاویر تولیدی را به تمیزدهنده نشان می‌دهیم، نه فقط آخرین وضعیت آن را. این مکانیزم باعث پایداری فرآیند آموزش شده و از نوسانات شدید در تابع زیان جلوگیری می‌کند.

### ۳.۱.۲ محدودیت‌های مدل

این مدل در وظایفی که نیازمند تغییرات هندسی بزرگ هستند (مانند تبدیل سگ به گربه یا سیب به موز که شکل کاملاً متفاوتی دارند) عملکرد ضعیفی دارد. دلیل اصلی، قید سازگاری چرخه‌ای ( $L_{cyc}$ ) است. این قید مدل را به شدت جریمه می‌کند اگر تصویر بازسازی شده با تصویر اصلی تفاوت داشته باشد. تغییرات هندسی بزرگ (مثل کشیدن یا جمع کردن تصویر) بازسازی دقیق تصویر اولیه را بسیار دشوار می‌کند. بنابراین، مدل ترجیح می‌دهد "مکان" پیکسل‌ها را تغییر ندهد و فقط "رنگ و بافت" آن‌ها را عوض کند.

## ۲.۲ پیاده‌سازی عملی

### ۱.۲.۲ آماده‌سازی داده‌ها

در این بخش از مجموعه داده apple2orange استفاده شده است. نمونه‌هایی از تصاویر این دیتا است در شکل ؟؟ نمایش داده شده است. همانطور که مشخص است، دامنه A شامل سیب‌ها و دامنه B شامل پرتقال‌ها می‌باشد.

### ۲.۲.۲ فرآیند آموزش

نکته مهم: در مقاله اصلی CycleGAN، مدل‌ها معمولاً برای ۲۰۰ اپیک آموزش می‌بینند. در این پروژه به دلیل محدودیت منابع پردازشی، آموزش تنها برای ۲۰ اپیک انجام شده است. همان‌طور که در ادامه تحلیل خواهد شد، نتایج فعلی امیدوارکننده هستند اما با ادامه آموزش، مولد می‌توانست در فریب دادن تمیزدهنده قوی‌تر عمل کرده و جزئیات ریزتری را تولید کند.

جدول ۱.۲: هایپرپارامترهای استفاده شده در آموزش مدل CycleGAN

پارامتر (Parameter)	مقدار (Value)	توضیحات (Value)
Batch Size	1	اندازه دسته (استاندارد (CycleGAN
Learning Rate	0.0002	نرخ یادگیری برای هر دو شبکه
Optimizer	Adam	بهینه‌ساز
$\beta_1$ Adam	0.5	پارامتر مومنتوم اول
$\beta_2$ Adam	0.999	پارامتر مومنتوم دوم
Epochs	20	تعداد کل دورهای آموزش
Image Size	$128 \times 128$	ابعاد تصاویر ورودی و خروجی
$\lambda_{cycle}$	10	ضریب اهمیت زیان سازگاری چرخه‌ای
$\lambda_{identity}$	5.0	ضریب اهمیت زیان همانی ( $0.5 \times \lambda_{cycle}$ )
Generator Arch	ResNet (6 blocks)	معماری مولد (برای تصاویر ۱۲۸ پیکسلی)
Discriminator Arch	PatchGAN	معماری تمیزدهنده (Patch 70 × 70)
Buffer Size	50	ظرفیت بافر تاریخچه تصاویر

### ۳.۲.۲ تحلیل خروجی‌های مدل

رای درک بهتر روند یادگیری مدل، خروجی‌های مدل در ایپاک‌های ۱، ۱۰ و ۲۰ را بررسی می‌کنیم.

**ایپاک ۱: شروع یادگیری** در شکل ۱.۲ نتایج اولیه نمایش داده شده است. همان‌طور که مشاهده می‌شود، تصاویر تولید شده تقریباً مشابه ورودی‌های اصلی هستند و یا حاوی نویزهای شدید و لکه‌های رنگی تصادفی می‌باشند (به عنوان مثال در تصویر پایین سمت راست، پس‌زمینه آبی به اشتباه به مشکی تبدیل شده است). در این مرحله، وزن‌های شبکه تصادفی است و تمیزدهنده هنوز تفاوت بین «پرتقال واقعی» و «پرتقال ساختگی» را یاد نگرفته است؛ لذا مولد بازخورد معناداری دریافت نمی‌کند. در اینجا زیان سازگاری چرخه‌ای غالب است و مدل را تشویق می‌کند تا نگاشت همانی (Identity Mapping) را انجام دهد.

**ایپاک ۱۰: شکل‌گیری رنگ‌ها** در شکل ۲.۲، تغییرات رنگی واضح‌تر شده‌اند:

- در تبدیل سیب → پرتقال، سطح قرمز سیب‌ها شروع به زرد یا نارنجی شدن کرده است.
- در تبدیل پرتقال → سیب، بافت پوست پرتقال صافتر شده و به قرمزی متمایل می‌شود.

با این حال، مدل هنوز دقت کافی ندارد و لبه‌های بین میوه و پس‌زمینه تار هستند. مدل توزیع کلی رنگ (Global Color Distribution) را یادگرفته اما در جزئیات ضعف دارد.

ایپاک ۲۰: نتیجه نهایی و ثبت بافت شکل ۳.۲ نتایج نهایی پروژه را نشان می‌دهد. ترجمه تصاویر بسیار پایدارتر و دقیق‌تر شده است:

- سیب‌ها به صورت یکنواخت نارنجی شده‌اند و بافت پوست پرتقال روی آن‌ها ظاهر شده است.
- حفظ پس‌زمینه: نکته بسیار مهم در تصویر بالا سمت راست این است که برگ‌های درخت سبز باقی مانده‌اند. این امر ثابت می‌کند که زیان سازگاری چرخه‌ای ( $\lambda_{cyc}$ ) و زیان همانی ( $\lambda_{id}$ ) به درستی عمل کرده‌اند و مدل موفق شده است «شیء» (میوه) را از «استایل دامنه» تقسیک کند.

#### ۴.۲.۲ تحلیل نمودارهای زیان

روند تغییرات توابع زیان در شکل ۴.۲ قابل مشاهده است.

- زیان سازگاری چرخه‌ای (منحنی سبز): این نمودار روندی کامل‌نزوی دارد. کاهش این خطأ تأیید می‌کند که نگاشتهای رفت و برگشت ( $x \approx F(G(x))$ ) در حال بهبود هستند و مدل یادگرفته است اطلاعات ساختاری لازم برای بازسازی تصویر را حفظ کند.
- زیان‌های تخاصمی (مولد و تمیزدهنده):
  - زیان تمیزدهنده ( $D_A, D_B$  - منحنی آبی) نوسان دارد یا به آرامی کاهش می‌یابد.
  - زیان مولد ( $G_A, G_B$  - منحنی نارنجی) نوسان دارد یا افزایش می‌یابد.

این عدم تقارن در GAN‌ها رایج است. تشخیص تصویر جعلی (به ویژه با وجود آرتیفکت‌ها) برای تمیزدهنده آسان‌تر از خلق تصویر بی‌نقص برای مولد است. در آموزش‌های کوتاه مدت، «برنده شدن» تمیزدهنده طبیعی است.

#### ۵.۲.۲ بخش امتیازی

در این بخش، مدل روی دیتاست Summer2Winter آموزش داده شد. نمونه داده‌های آموزشی در شکل ۵.۲ آورده شده است.

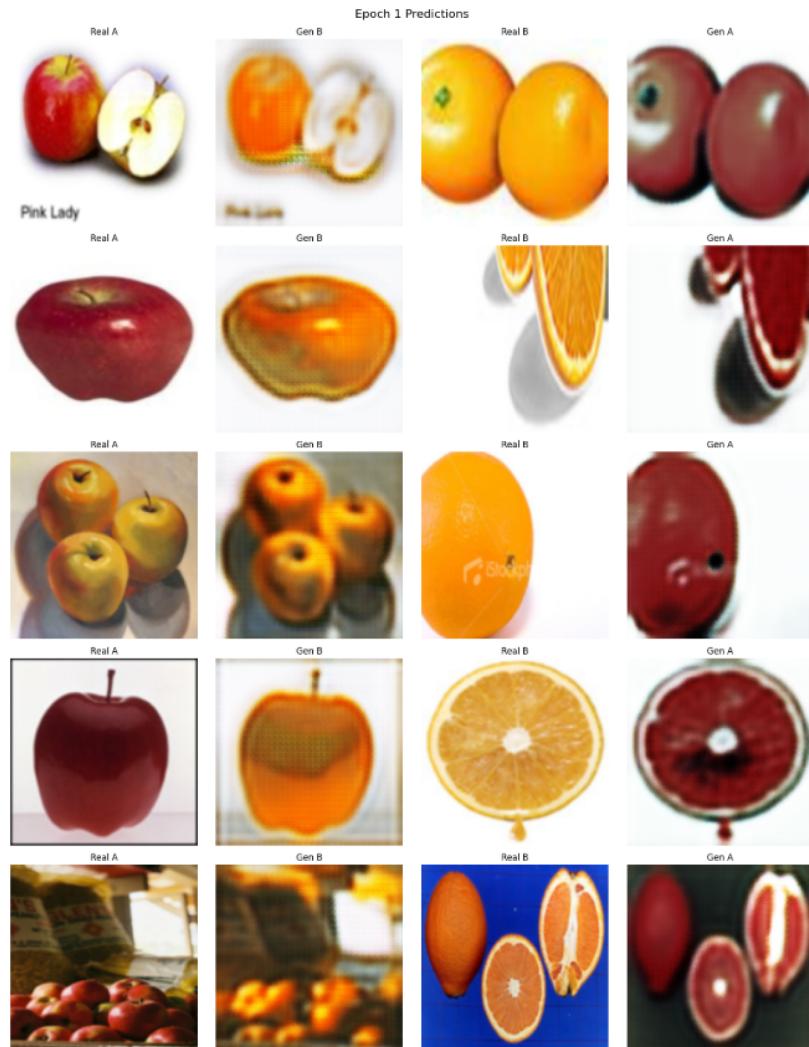
تحلیل خروجی‌های مدل روند پیشرفت تصاویر از ایپاک ۱ تا ۲۰ (که در شکل‌های ۶.۲ تا ۸.۲ نمایش داده شده است) رفتارهای یادگیری خاصی را آشکار می‌کند:

- جایگزینی بافت: بارزترین ویژگی، جایگزینی شاخ و برگ سبز با بافت‌های برفی سفید است. تا

- ایپاک ۲۰، مدل با موفقیت نواحی «چمنی» را شناسایی کرده و آنها را سفید/خاکستری می‌کند.
- قطعه‌بندی معنایی ضمیمی: مدل بین «آسمان» (که آبی/خاکستری می‌ماند) و «زمین» (که سفید می‌شود) تمایز قائل می‌شود که نشان‌دهنده عملکرد صحیح مدل در درک محتوای تصویر است.
  - حدودیت مدل: در جنگل‌های متراکم تابستانی، مولد «زمستان» گاهی به جای حذف کامل برگ‌ها، صرفاً یک فیلتر سفید ساده روی آنها اعمال می‌کند. این یک محدودیت شناخته‌شده در CycleGAN است؛ چراکه این مدل برای انتقال بافت (Texture-transfer) طراحی شده است، نه اصلاح هندسی (Geometry-modification).
  - آرتیفکت‌ها: برخی مصنوعات شطرنجی (Checkerboard artifacts) ممکن است در نواحی آسمان دیده شوند که ناشی از لایه‌های ConvTranspose2d در معماری مولد است.

#### تحلیل نمودارهای زیان با توجه به شکل ۹.۲:

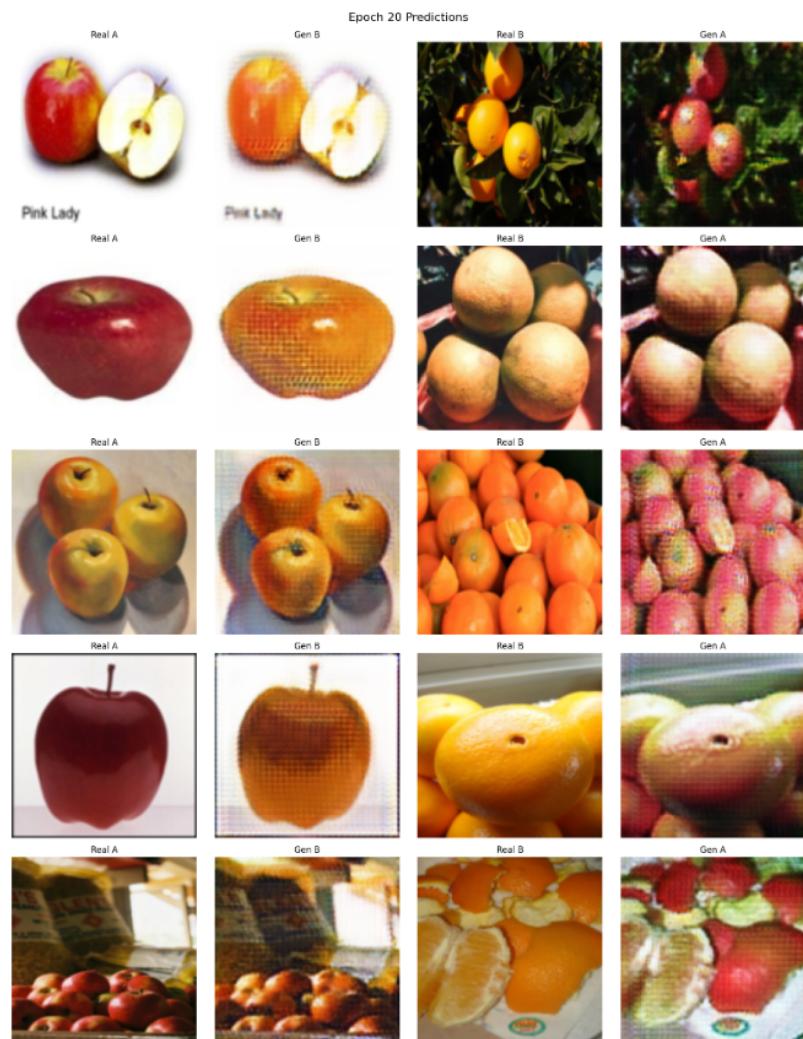
- روندهای زیان چرخه‌ای: کاهش مداوم این زیان تأیید می‌کند که نگاشتهای  $G$  و  $F$  به طور وارون پذیر (Bijective) می‌شوند. مدل یاد می‌گیرد اطلاعات «تابستان» را درون تصویر «زمستان» کدگذاری کند تا بتواند آن را بازسازی نماید؛ به همین دلیل جزئیات ساختاری کوه‌ها و صخره‌ها حفظ شده است.
- زیان تخاصمی: نوسان این زیان‌ها نشان‌دهنده یک بازی پایدار «مینیمم-ماکزیمم» (min-max) است که در آن هیچ‌یک از شبکه‌ها بر دیگری غلبه کامل نمی‌کند.



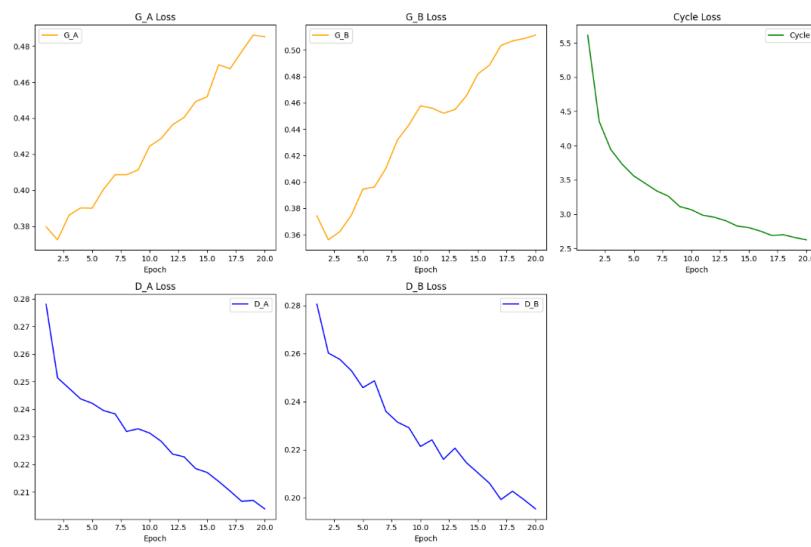
شکل ۱.۲: ایپاک ۱ برای دیتاست Apple2Orange



شکل ۲.۲: ایپاک ۱۰ برای دیتاست Apple2Orange

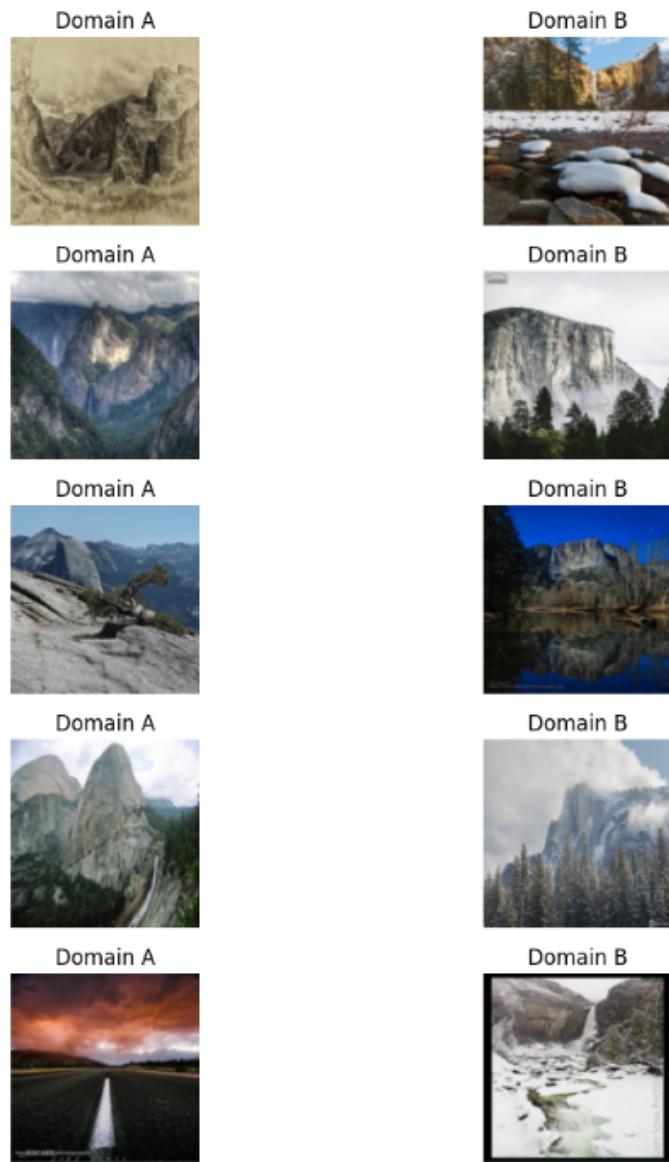


شکل ۳.۲: ایپاک ۲۰ برای دیتاست Apple2Orange



شکل ۴.۲: نمودارهای زیان برای دیتاست Apple2Orange

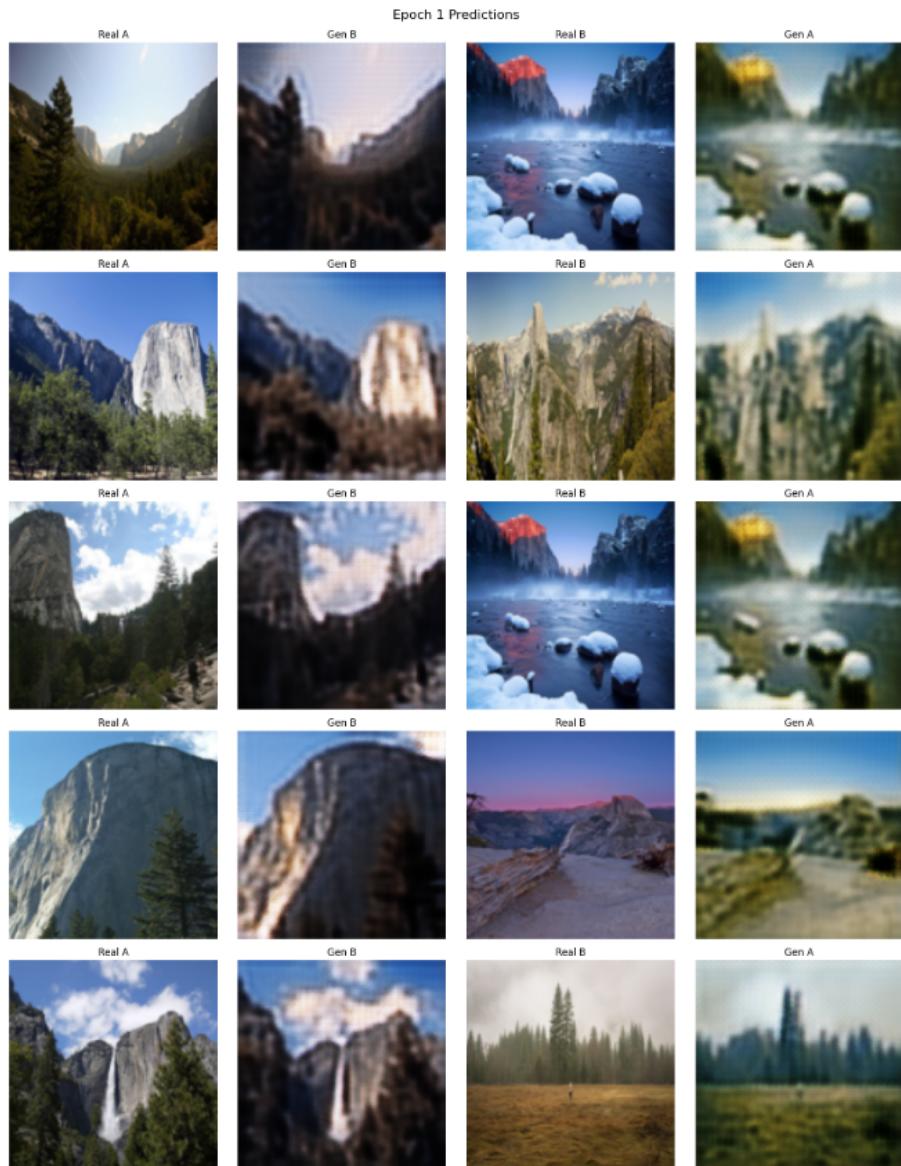
### Raw Dataset Samples (Before Training)



شکل ۵.۲: نمونه تصاویر دیتاست Summer2Winter

۲.۲. پیاده‌سازی عملی

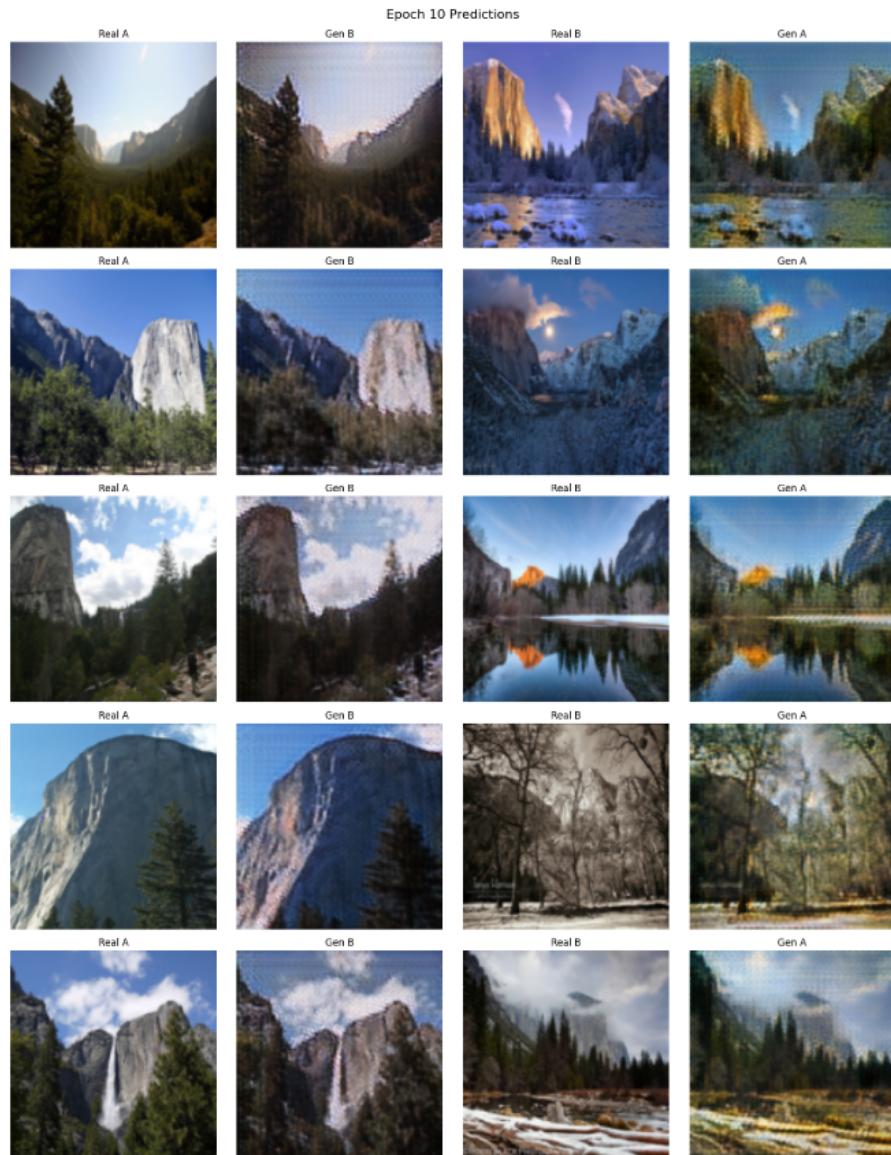
۲۹



شکل ۶.۲: ایپاک ۱ برای دیتاست Summer2Winter

## ۲.۲. پیاده‌سازی عملی

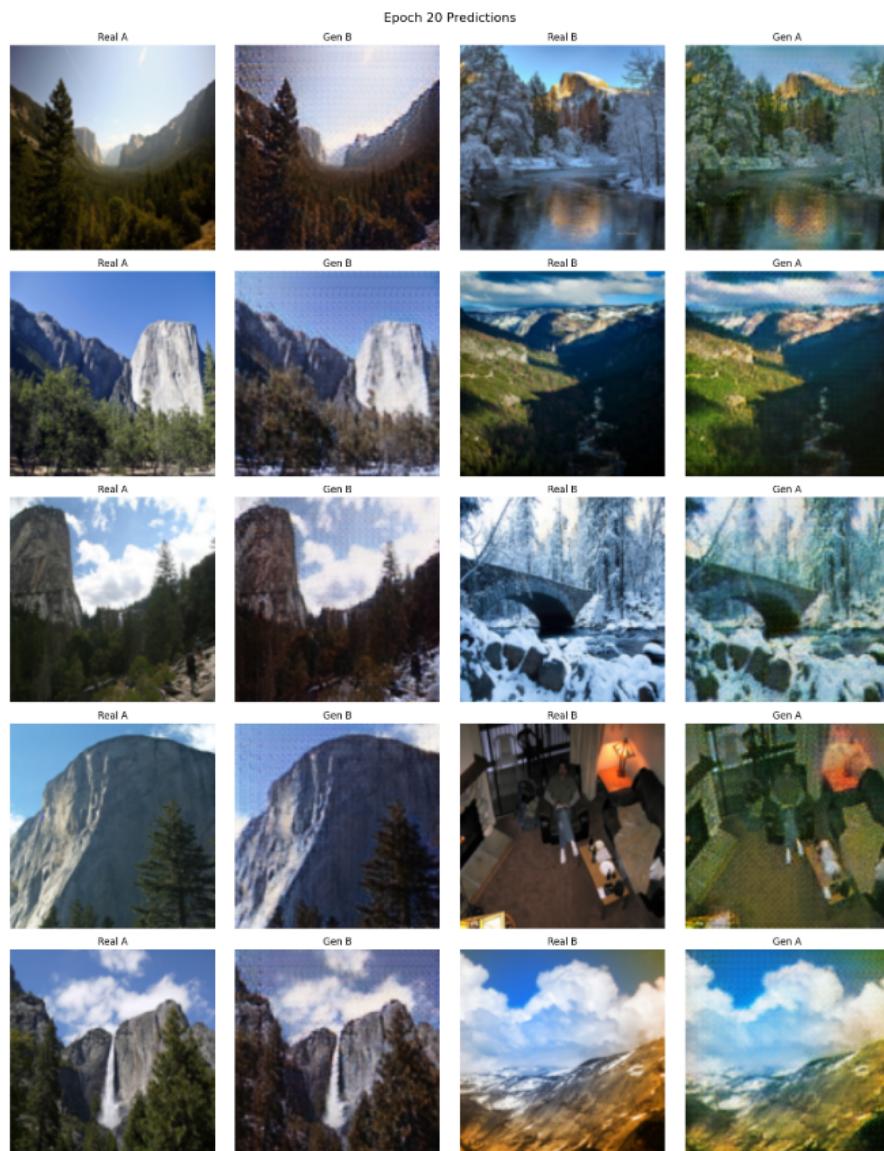
۳۰



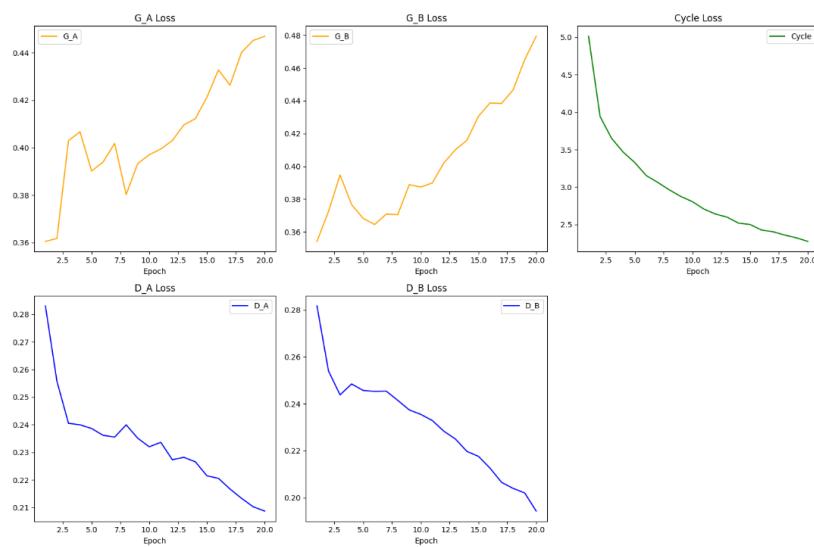
شکل ۷.۲: ایپاک ۱۰ برای دیتاست Summer2Winter

## ۲.۲. پیاده‌سازی عملی

۳۱



شکل ۸.۲: ایپاک ۲۰ برای دیتاست Summer2Winter



شکل ۹.۲: نمودارهای زیان برای دیتاست Summer2Winter