**1.1.a**

در مدل SimCLR از لاس contrastive استفاده می‌شود. طبق این objective زوج‌های مثبت باید به هم نزدیک شوند و زوج‌های منفی نیز باید از هم دور شوند. اهمیت دور کردن زوج‌های منفی از هم این است که در صورت عدم وجود آن مدل بهینه‌های محلی بدی می‌تواند پیدا کند مثلاً می‌تواند تمام ورودی‌ها را به یک نقطه map کند و همچنان لاس خوبی بگیرد.

مدل MoCo نیز از همین لاس استفاده می‌کند و بنابراین در مخرج لاس دور شدن نمونه‌های منفی را دارد و این زوج‌های منفی مهم هستند. اما برای نگهداری این زوج‌های منفی از یک صف استفاده می‌کند تا مشکل batch size بزرگ را نداشته باشد. (نمونه‌های داخل صف را توسط شبکه‌ی momentum enc ایجاد می‌کند)

در مدل BYOL اما از لاس contrastive استفاده نمی‌شود و فقط نمونه‌های مثبت به هم نزدیک می‌شوند. BYOL از 2 شبکه جدا به نام‌های online و target استفاده می‌کند. شبکه‌ی online یکی از augment های ورودی را امبد می‌کند و شبکه target یکی دیگر از augment ها را. لاس به این صورت تعریف می‌شود که خروجی دو شبکه نزدیک هم باشد یعنی به بیان دیگر شبکه online بتواند شبکه target را پیش‌بینی کند.
نیازی آپدیت شدن شبکه‌ی target توسط گرادیان نیست و moving avg شبکه online است که باعث می‌شود بهینه‌های محلی ذکرشده پیش نیاید و نیازی به زوج‌های منفی نباشد

**1.1.b**

آموزش مدل با batchsize بزرگ دقتی از SGD/Momentum عادی استفاده شود unstable است. برای stability بیشتر از LARS استفاده می‌شود. همچنین برای batch size های کوچک‌تر نشان داده شده که اسکیل کردن learning rate به اندازه جذر batch size کمک کننده است.
همچنین از global batch norm استفاده شده است. به این معنا که batchnorm را به صورت global در سطح تمام device ها محاسبه می‌کنیم نه per device.
این کار باعث می‌شود مدل برای یادگیری بازنمایی‌ها نتواند shortcut بزند

**1.2.a**

اگر پارامترهای مدل teacher را $\theta_t$ و پارامترهای مدل student را $\theta_s$ بنامیم، مدل teacher از طریق گرادیان آپدیت نمی‌شود و به صورت moving average از روی شبکه student به دست می‌آید یعنی $\theta_t = m\theta_t + (1-m)\theta_s$ که در آن m بسیار نزدیک 1 است.
تعبیری که می‌توان برای شبکه teacher ارائه داد آن است که می‌توان آن را به صورت ensemble چندین مدل دانست. در حقیقت این شبکه ensemble وزن‌های نمایی از student است و از هرکدام آن‌ها بهتر است ← می‌تواند شبکه student را هدایت کند

**1.2.b**

به ازای هر تصویر دو کراپ global و چندین کراپ local داریم. تنها global view ها به شبکه teacher ورودی داده می‌شوند و باعث ایجاد local to global correspondence در آن می‌شوند. به این صورت شبکه teacher سعی دارد امبدینگ‌های سازگاری برای این کراپ‌ها به دست آورد، بنابراین به اشتراک‌های این دو کراپ global توجه بیشتری می‌کند و از آن جا که object اصلی در هر دو کراپ global وجود دارد شبکه نیز به آن توجه بیشتری می‌کند تا در امبدینگ لحاظ شود

## 1.2.c

به طور کلی از روش های مختلفی برای جلوگیری از collapse استفاده می‌شود مانند contrastive loss , BN , ... اما در این روش استفاده از تکنیک های centering و sharpening بر روی خروجی های شبکه teacher برای جلوگیری از collapse کافی است.

تکنیک centering مانع آن می‌شود که یک بُعد سایر ابعاد را dominate کند اما مدل را ترغیب می‌کند که همه ابعاد خروجی اش شبیه uniform شوند.

تکنیک sharpening نیز برعکس آن را انجام می‌دهد. بنابراین استفاده از هر دوی آنها تاثیرات شان را بالانس می‌کند و باعث می‌شود مدل collapse نکند.

تکنیک centering معادل اضافه کردن یک بایاس به نتیجه شبکه teacher است و sharpening نیز با انتخاب مقداری کوچک برای $\tau_t$ در نرمالیزیشن softmax شبکه teacher انجام می‌شود

## 1.2.d

efficient implementation

- fast and memory efficient attention

نسخه جدیدی از Flash Attention پیاده سازی شده که از قبل بهتر است و همچنین هایپر پارامتر هایی نیز برای استفاده بهینه عنوان شده است:

embedding dim = 1536 , number of heads = 24 vit-g backbone has 1.1B parameters

- sequence packing

یک تکنیک NLP است که در آن sequence های حاصل از همه کراپ های مختلف باهم concat می‌شوند و به مدل ورودی داده می‌شوند. این برخلاف DINO است که مجبور بود برای هر کراپ این کار را به صورت جدا انجام دهد. همچنین برای اینکه کراپ های مختلف روی هم تاثیر نداشته باشند از ماسک های block diagonal استفاده می‌شود و بهبود سرعت قابل توجهی حاصل می‌شود

- efficient stochastic depth

محاسبه dropped residuals که در نسخه قبلی داشتیم اسکیپ می‌شود به جای آنکه محاسبه شده و نتایج mask شوند. این کار حافظه و توان پردازشی را save می‌کند

- fully sharded data parallel (FSDP)

پارامترهای 4 مدل teacher, student, optimizer first moments, optimizer second moments بر روی 4 تا GPU تقسیم شده است. در مدل قبلی روی یک دستگاه بودند که باعث می‌شد GPU ها 16GB نیاز باشد. این کار توسط پیاده سازی FSDP در pytorch انجام شده.

- Model distillation

به جای آموزش مدل های کوچکتر from scratch ، آنها را از distill شده مدل بزرگ Vit-g به دست می‌آوریم

1.2.e

- در BYOL از لاس MSE استفاده می‌شود اما در DINO از cross entropy استفاده می‌شود.
  ضمناً در DINO قبل از اعمال لاس مرزهی ها sharpen , center می‌شوند

- در DINO از backbone های مبتنی بر Vit استفاده شده اما در BYOL از ResNet

- در DINO از تکنیک multi crop استفاده شده

## 2.1.a

برای permutation equivarient بودن باید وزن همسایه ها به صورت shared باشد. این طوری مدل نمی تواند یاد بگیرد که به برخی همسایه ها وزن بیشتری نسبت دهد. در این بخش نیز وزن همسایه ها همگی $W_2$ است پس معتبر است

## 2.1.b

این حالت معتبر نیست زیرا وزن های متفاوتی برای همسایه ها درنظر گرفته شده

## 2.1.c

این حالت معتبر است زیرا وزن تمام همسایه ها $W_2$ است. ضمناً می دانیم که ماکسیمم گرفتن سولفه‌ای نیز ترتیبی ایجاد نمی کند زیرا خود max نسبت به ترتیب آرگومان هایش permutation invarient است

## 2.2

ابتدا فرم ماتریسی ارسال پیام را می نویسیم :

$$H^{(t+1)} = \sigma\left(A H^{(t)} W' + H^{(t)} W\right)$$

عبارت اول امبدینگ همسایه ها را تجمیع می کند و عبارت دوم امبدینگ خود نود. در حقیقت برای عبارت دوم داریم:

$$H W = \begin{bmatrix} h_1^T W \\ \vdots \\ h_n^T W \end{bmatrix}$$

یعنی امبدینگ هر نود در ماتریس W ضرب می شود و در سطر ماتریس جدید قرار می گیرد

حال یک جایگشت روی نود های گراف اعمال می کنیم و آنرا به GNN ورودی می دهیم. برای این کار باید سطر و ستون های ماتریس مجاورت $(A)$ جایگشت پیدا کنند و همچنین سطر های ماتریس $H^{(t)}$ نیز همان جایگشت را داشته باشند پس به جای A باید $PAP^T$ را قرار دهیم، به جای $H^{(t)}$ باید $PH^{(t)}$ را قرار دهیم:

$$\sigma\left(PA\underbrace{P^T P}_{I}H^{(t)} W' + PH^{(t)} W\right)$$

$$= P\sigma\left(AH^{(t)}W' + H^{(t)}W\right) = PH^{(t+1)}$$

می دانیم که ماتریس های permutation، unitary هستند پس $P^T = P^{-1}$

بنابراین اثبات شد که اگر ماتریس را جایگشت دهیم و به GNN ورودی بدهیم معادل آن است که ماتریس اولیه را به GNN بدهیم و نتیجه آن را جایگشت دهیم

permutation equivarient

## 3.1

آشفتگی متخاصمانه فراگیر نوع خاصی از آشفتگی متخاصمانه است که image agnostic است. تفاوت آن با حالت عادی آن است که یک آشفتگی کوچک به صورت ثابت به تمام ورودی ها اعمال می‌شود و نیازی نیست که برای هر ورودی، به صورت جدا آشفتگی محاسب شود. بنابراین این نوع آشفتگی که magnitude کوچکی دارد و توسط چشم انسان قابل رویت نیست به تعداد ورودی اعمال شده و باعث می‌شود مدل دسته بند، کلاس اشتباهی را به عنوان کلاس آن تصویر اعلام کند.

## 3.2

یافتن آشفتگی‌ها از منظر تأمین امنیت کاربران مدل و به طور کلی robust کردن مدل بسیار حائز اهمیت است. به خصوص این نوع آشفتگی به دلیل universal بودن آن می‌تواند بسیار خطرناک باشد و همان طور که در مقاله اشاره شده است به یازده‌ی وسیعی از مدل‌ها نیز apply می‌شود. فرض کنید یک خودروی خودران می‌خواهد تابلو های پیش روی خود را شناسایی کند. با این روش می‌توان یک آشفتگی فراگیر و ثابت یافت و به هر تصویر ورودی مدل اضافه کرد. به این صورت مدل باعث ایجاد صدمات قابل ملاحظه ای می‌شود.

همچنین از دید تکنیکال نیز می‌توان گفت وجود چنین آشفتگی های فراگیری نشان دهنده‌ی وجود correlation های هندسی مهمی در مرز تصمیم‌گیری دسته‌بند ها در ابعاد بالا است و اینکه جهت هایی در فضای ورودی وجود دارند که تنها با حرکت در آن راستا (صرف نظر از ورودی) می‌توان دسته بند را فریب داد.

## 3.3

we want to find $v$ that satisfies:

- $\|v\|_p \leq \xi$
- $\mathbb{P}_{x \sim \mu}\left(\hat{k}(x+v) \neq \hat{k}(x)\right) \geqslant 1-\delta$ ← Fooling rate

یعنی احتمال فریب خوردن مدل $k$ برروی داده هایی که از توزیع $\mu$ می‌آیند از $1-\delta$ که آنرا تعیین می‌کنیم، بیشتر باشد. مقید به آنکه اندازه آشفتگی نیز از $\xi$ کمتر بماند

به این منظور الگوریتم زیر را خواهیم داشت:

$$X_v \triangleq \{x_1+v \dots x_m+v\}$$
$$Err(X_v) \triangleq \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}_{\hat{k}(x_i+v) \neq \hat{k}(x_i)}$$
$$P_{p,\xi}(v) \triangleq \underset{v'}{\operatorname{argmin}} \|v-v'\|_2 \quad \text{s.t.} \quad \|v'\|_p \leqslant \xi$$

initialize $v \leftarrow 0$

while $Err(X_v) \leqslant 1-\delta$:

  for $x_i \in X$:

    if $\hat{k}(x_i+v) = \hat{k}(x_i)$:

      $\Delta v_i \leftarrow \underset{r}{\operatorname{argmin}} \|r\|_2 \quad \text{s.t.} \quad \hat{k}(x_i+v+r) \neq \hat{k}(x_i)$

  $v \leftarrow P_{p,\xi}(v + \nabla v_i)$

return $v$

به این معنی که روی دیتاست $X$ حرکت می‌کنیم و به ازای هر $x_i$ که مدل با ورودی $x_i+v$ فریب نخورده بود، $\nabla v_i$ ای بدست می‌آوریم که با اضافه کردن آن به $v$، مدل فریب بخورد. برای ارضای قید $\|v\|_p \leq \xi$ بردار $v+\nabla v_i$ را برروی دایره $l_p$ به شعاع $\xi$ پروجکت می‌کنیم و $v$ را با نتیجه آن به روز رسانی می‌کنیم

⊛ تابع $f$ که در صورت سوال به آن اشاره شده، حداقل احتمال موفقیت حمله را نشان می‌دهد و معادل $1-\delta$ می‌باشد. دیتاست را نیز به جای $D$، $X$ گرفتیم.

## 4.1

هر 3 مدل سعی دارند که multi modal learning انجام دهند و بتوانند امبدینگ های مشترک متن و تصاویر ایجاد کنند.
CLIP سعی می‌کند یک multi modal joint embedding به دست بیاورد که در آن تصاویر و متن های مشابه، به هم نزدیکتر باشند. SimVLM نیز در بخش enc خود سعی دارد امبدینگ مناسبی برای متن و تصویر ورودی به دست بیاورد تا در بخش dec بتواند با توجه به آن امبدینگ مشترک، متن تولید کند. Coca نیز توسط لاس contrastive loss امبدینگ های حاصل از img enc و txt dec را نزدیک نگه می دارد

هر 3 مدل قابلیت Zero shot learning را دارند و می‌توان به عنوان مثال Zero shot classification انجام داد.
CLIP و Coca که هردو قابلیت dual enc دارند، در درس راجع به آن صحبت شد. SimVLM نیز می‌تواند به طریق مشابه تمایلیت ای برای تبدیل کلاس ها به متن در نظر بگیرد و تصویر و آن متن را به بخش enc ورودی دهد. حال می‌توان از متن generate شده، تعلق به آن کلاس را فهمید.

تفاوت ها

این 3 مدل در pretraining objective باهم تفاوت دارند. CLIP با contrastive pretraining سعی می‌کند pair های متن و تصویر مشابه را به هم نزدیک و از بقیه دور کند. SimVLM از prefix language model objective استفاده می‌کند که سعی می‌کند با توجه به امبدینگ های متن و تصویر ورودی، متن تولید کند. Coca اما این دو ایده را ترکیب می‌کند، و علاوه بر اینکه با contrastive loss امبدینگ های متن، تصویر را align می‌کند، متن نیز برای آن تولید می‌کند.

از نظر کاربرد نیز این 3 مدل قادر به حل کردن تسک های متفاوتی هستند

| | CLIP | SimVLM | Coca |
|---|---|---|---|
| txt to img retrieval | ✓ | | ✓ |
| img to txt retrieval | ✓ | | ✓ |
| img captioning | | ✓ | ✓ |
| VQA | | ✓ | ✓ |

## 4.2.a

در بخش txt enc از transformer استفاده شده است. به طور دقیق‌تر می‌توان گفت که ماژول ترنسفورمر آن 12 لایه با عرض 512 دارد و شامل 8 تا attention head است.

برای به‌دست آوردن امبدینگ متن ابتدا از ترنسفورمر استفاده می‌شود و سپس یک لایه projection خطی روی آن اعمال می‌شود تا به فضای امبدینگ مشترک برسیم.

در بخش img enc از دو معماری استفاده می‌شود. یکی ResNet 50 که کمی تغییر یافته و دیگری VIT. بسته به کاربرد می‌توان از هرکدام استفاده کرد. در این بخش نیز پس از img enc یک لایه projection خطی روی آن اعمال می‌شود تا به فضای امبدینگ مشترک برسیم.

در نهایت در فضای مشترک لاس contrastive اعمال می‌شود

## 4.2.b

$$\frac{\partial s_{ij}}{\partial x_i} = \frac{y_j \|x_i\| \|y_j\| - \|y_j\| \frac{x_i}{\|x_i\|} (x_i \cdot y_j)}{\|x_i\|^2 \|y_j\|^2} = \frac{\|x_i\|^2 y_j - (x_i \cdot y_j) x_i}{\|x_i\|^3 \|y_j\|}$$

$$\frac{\partial L_i}{\partial x_i} = \frac{-1}{N} \frac{\sum_j e^{s_{ij}/\tau}}{e^{s_{ii}/\tau}} \left[ \left( \frac{1}{\tau} e^{s_{ii}/\tau} \frac{\partial s_{ii}}{\partial x_i} \right) \sum_j e^{s_{ij}/\tau} - \left( \sum_j \frac{1}{\tau} e^{s_{ij}/\tau} \frac{\partial s_{ij}}{\partial x_i} \right) e^{s_{ii}/\tau} \right]$$

$$\times \frac{1}{\left( \sum_j e^{s_{ij}/\tau} \right)^2}$$

$$= \frac{(x_i \cdot y_i) x_i - \|x_i\|^2 y_i}{N\tau \|x_i\|^3 \|y_i\|} + \frac{\sum_j \left( e^{s_{ij}/\tau} \frac{\|x_i\|^2 y_j - (x_i \cdot y_j) x_i}{\|x_i\|^3 \|y_j\|} \right)}{N\tau \sum_j e^{s_{ij}/\tau}}$$

در برخی سوال‌ها با علی بابایگ همفکری داشتم