Реферат

У роботі проводиться розрахунок площі криволінійного паркану змінної висоти. Висота задається як функція декартових координат точки паркана.

Пошук алгоритмів розв'язання задачі виконаний на сайті [?]. Для обчислення площі використовуються криволінійні інтеграли 1-го роду [?]. Чисельні приклади розв'язані у MATLAB [?]. Використовувалася також сторінка [?].

Іл. 0. Табл. 0. Лист. 0. Бібліогр.: 0 назв.

ЗМІСТ

ПЕРЕЛІК ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

АКФ – автокореляційна функція;

ДР – дипломна робота;

ЕОМ – електронна обчислювальна машина;

КПО – коефіцієнт природного освітлення;

НДР – науково-дослідна робота;

AR – autoregression;

ARMA – autoregression moving average;

ARIMA – autoregression integrated moving average;

DOTS – Docker-oriented testing system;

MA – moving average;

MAPE – mean absolute percentage error;

SSA – singular spectrum analysis;

SVD – singular value decomposition.

ВСТУП

З кожним роком все більшу роль відіграють статистичні методи та алгоритми для аналізу різноманітних даних. Метою даної дипломної роботи є дослідження часового ряду тестувальної системи DOTS для подальшого прогнозу.

Тестувальна система DOTS (Docker-oriented testing system) застосовується для автоматизації тестування алгоритмічних задач і є найбільшою автоматизованою системою перевірки задач з інформатики у харківській області, та однією з найбільших в Україні. Щорічно на базі цієї системи проходят олімпіади, турніри та різноманітні курси. І, з кожним роком, кількість користувачів зростає, а з ними і кількість серверів необхідних для нормального функціонування системи. І було б дуже доречно вміти передбачати кількість потрібних серверів на наступний день, щоб їх не було і замало, коли користувачі відчувають дискомфорт через занадто довгі черги задач, і щоб їх не було занадто багато, і не доводилось платити за сервери, які не працюють.

В даній роботі досліджуються дані, які складаються з числа щоденних відправлень завантажених рішень. Саме в цьому мають стати в нагоді методи з аналізу часових рядів.

Для аналізу структури ряду та побудови прогнозу в дипломній роботі були використані методи ARIMA (autoregressive integrated moving average) та SSA (singular spectrum analysis).

За умови їх використання були вирішені наступні задачі:

1) підготовка вихідних даних;
2) аналіз структури часового ряду;
3) аналіз точності побудованих моделей (ARIMA);
4) прогнозування;
5) порівняння роботи методів.

1 ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ

Термін самонавчання (self-supervised learning, SSL) використовувався у різних контекстах та сферах, таких як навчання уявлень (representation learning), нейронні мережі, робототехніка, обробка природних мов (natural language processing, NLP) та навчання з посиленням (reinforcement learning). У всіх випадках основна ідея полягає в тому, щоб автоматично генерувати якийсь наглядовий сигнал для вирішення якогось завдання (як правило, вивчати подання даних або автоматично мітити набір даних).

В типовій задачі SSL є величезна нерозмічена вибірка, необхідно сформувати для кожного об'єкта псевдо-мітку (pseudo label) і вирішити отриману SSL-завдання, але нас цікавить не стільки якість рішення придуманого нами завдання (її називають pretext task), скільки представлення (representation) об'єктів, яке буде вивчено в процесі її рішення. Це представлення можна в подальшому використовувати вже при вирішенні будь-якої задачі з мітками (SSL), яку називають наступною задачею (downstream task). Одна з головних причин самонавчання — невеликий обсяг розмічених даних. На відміну від навчання з частково розміченими даними в самонавчанні використовуються абсолютно довільні нерозмічену дані, що не мають відношення до розв'язуваної задачі.

Сучасна обробка тексту (NLP) приблизно на 80% складається з самонавчання. Наприклад, за допомогою самонавчання можна знайти майже всі представлення слів (а також текстів). Наприклад, в класичному алгоритмі word2vec беруть нерозмічений корпус текстів, потім самі придумують завдання з мітками (по сусіднім словами передбачити центральне слово або, навпаки, по центральному сусідні з ним), навчають на цьому завданні просту нейронну мережу, в результаті виходять векторні уявлення слів, які вже використовуються в інших, ніяк не пов'язаних з попередньою, задачах. Часто такий підхід називають також трансферним навчанням (transfer learning). Вцілому, трансферне навчання більш широке поняття - коли модель, навчену для вирішення однієї задачі, використовують при вирішенні іншої. У самонавчанні важливо, щоб розмітка в попередньої задачі (псевдо-мітки) виходила автоматично. Наприклад, представлення, отримані методом

Cove, в якому використовується кодер для завдання машинного перекладу, будуть трансферними, але не отриманими за допомогою самонавчання.

До основних термінів можна віднести:

1) Попередня задача (pretext task) — задача з штучно створеними мітками (псевдо-мітками), на якій навчається модель, щоб вивчити хороші вистави (representations) об'єктів. Наприклад, є надія, що в нейронної мережі на попередньої задачі навчаться початкові і середні шари, їх потім заморожують і навчають останні шари.
2) Псевдо-мітки (pseudo labels) — мітки, які виходять автоматично, без ручного розмітки, але навчання яким сприяє формуванню хороших представлень.
3) Подальша задача (downstream task) — задача на якій перевіряють якість отриманих уявлень. Майже у всіх експериментах в статтях по самонавчання на отриманих ознакових представленнях в наступних завданнях навчають прості моделі: логістичну регресію або метод найближчого сусіда. Таким чином, самонавчання - це напрямок в глибокому навчанні, який прагне зробити глибоке навчання процедурою попередньої обробки даних, тобто мережі потрібні для формування ознак, вони навчаються на дешевій розмітці великих наборів спочатку нерозмічену даних, а сама задача вирішується простою моделлю.

1.1 Порівняльне навчання

В останні роки значно менше робіт з вибору попередніх завдань, оскільки основним напрямком в самообучении стало порівняльне навчання (Contrastive Learning). На вхід нейромережі подається пара об'єктів і вона визначає, схожі вони чи ні. Якщо об'єкт подається як аугментований патч з зображення, схожі повинні бути патчі з одного зображення, а не схожі — з різних. В якості оптимізуємих функцій використовується взаємна інформація або пов'язані з нею функції, наприклад, її нижня оцінка InfoNCE :

$$-E_x \left[\log \frac{exp(f(x)^T f(x^+))}{exp(f(x)^T f(x^+)) + \sum_{j=1}^{N-1} exp(f(x)^T f(x_j))} \right], \tag{1.1}$$

де x – обраний патч, званий якірним,

x^+ – схожий на нього,

x_j – несхожий (їх $N-1$ штука),

$f()$ – кодер об'єкту (подання його у вигляді речового вектора).

Така функція помилки майже перетворюється в Triplet Loss, якщо використовувати тільки 1 негативний приклад.

1.2 Deep InfoMax

Окреслимо загальні параметри навчання кодеру для максимізації взаємної інформації між його входом і виходом. Нехай X і Y є областю і діапазоном неперервної та (майже скрізь) диференційованої параметричної функції, $E_\psi : X \to Y$ з параметрами ψ (наприклад, нейронна мережа). Ці параметри визначають сімейство кодерів $E_\Psi = E_{\phi_{\psi \in \Psi}}$ над Ψ. Припустимо, що нам дано набір навчальні приклади на вхідному просторі, X: $\mathbf{X} := x^{(i)} \in X_{i=1}^N$, з емпіричним розподілом ймовірностей $\mathbb{P}$. Визначимо $\mathbb{U}_{\psi,\mathbb{P}}$ як граничний розподіл, індукований проштовхуванням зразків від $\mathbb{P}$ через $\mathbb{E}_\psi$. Тобто, $\mathbb{U}_{\psi,\mathbb{P}}$ – розподіл за кодуваннями $y \in Y$, отриманий шляхом вибірки спостережень $x \sim X$, а потім вибірки $y \sim E_\psi(x)$.

Приклад кодера даних зображень наведено на рис. 1.1, який буде використаний у наступних розділах, але цей підхід можна легко адаптувати для тимчасових даних.

Подібно до принципу оптимізації infomax, можна стверджувати, що кодер повинен навчатися відповідно до таких критеріїв:

1) Максимізація взаємної інформації. Необхідно знайти набір параметрів ψ, таких, щоб взаємна інформація $L(X; E_\psi(X))$ була максимальною. Залежно від кінцевої

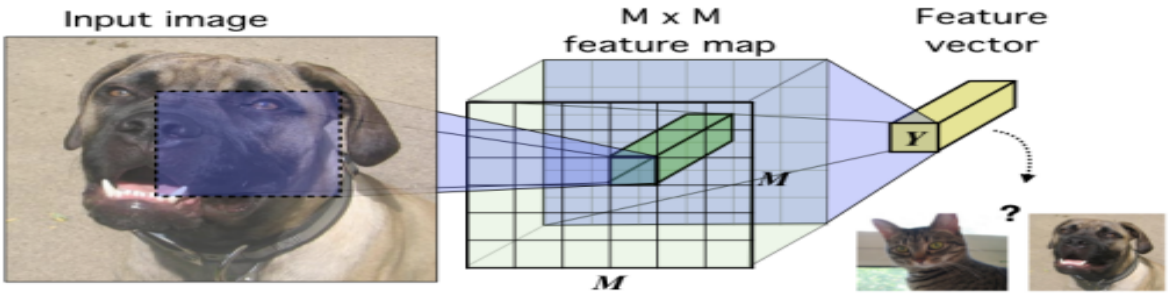

Рисунок 1.1 – Базова модель кодера в контексті даних зображень.

цілі, це максимізація може бути здійснена за повним входом, X або деякою структурованою або "локальною" підмножиною.

2) Статистичні обмеження. Залежно від кінцевої мети для подання, граничні $\mathbb{U}_{\psi,\mathbb{P}}$ повинні відповідати попередньому розподілу, $\mathbb{V}$. Грубо кажучи, це може бути використано для заохочення вихідних даних кодера до бажаних характеристик (наприклад, незалежності).

Формулювання цих двох цілей називається Deep InfoMax (DIM).

Основні рамки максимізації взаємної інформації представлені на рис. 1.2.

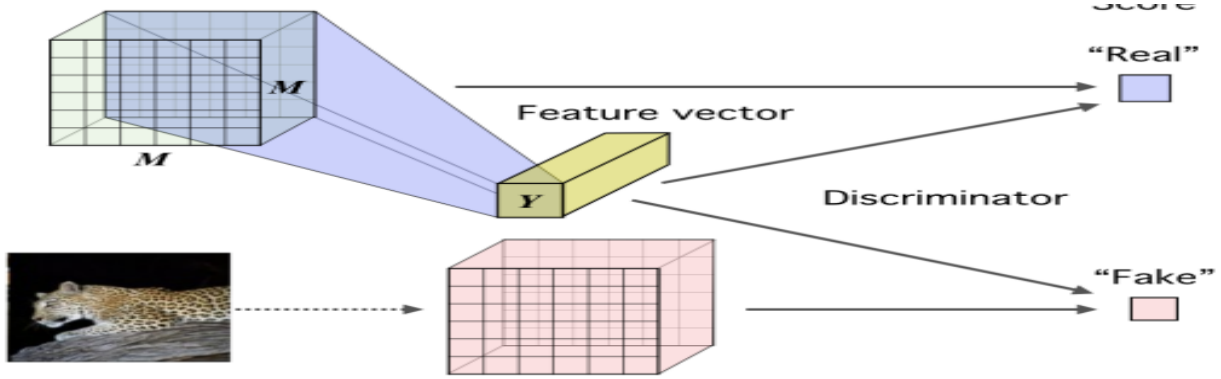

Рисунок 1.2 – Deep InfoMax (DIM) з глобальною MI(X; Y) ціллю.

Підхід слідує взаємній інформаційній нейронній оцінці (MINE), яка оцінює взаємну інформацію шляхом підготовки класифікатора для розрізнення зразків, що надходять із спільного розподілу $\mathbb{J}$ та добутку граничних значень, $\mathbb{M}$, випадкових величин X та Y. MINE використовує нижню межу для MI на основі представлення Донскер-Варадхана (DV) KL-дивергенції,

$$L(X;Y) := D_{KL}(\mathbb{J}||\mathbb{M}) \geq \hat{L}_{\omega}^{DV}(X;Y) := \mathbb{E}_{\mathbb{F}}|T_{\omega}(x,y)| - \log \mathbb{E}_{\mathbb{M}}[e^{T_{\omega}(x,y)}], \quad (1.2)$$

де $T_{\omega} : X \times Y \to \mathbb{R}$ функція дискримінатора, змодельована нейронною мережею з параметрами ω.

На найвищому рівні оптимізується $\mathbb{E}_{\psi}$ одночасно оцінюючи та максимізуючи $L(X, E_{\psi}(X))$,

$$(\hat{\omega}, \hat{\psi})_G = \arg\max_{\omega,\psi} \hat{L}_{\omega}(X; E_{\psi}(X)), \quad (1.3)$$

де нижній індекс G позначає «глобальний». Однак існують деякі важливі відмінності, які відрізняють цей підхід від MINE. По-перше, оскільки кодер і оцінювач взаємної інформації оптимізують одну і ту ж мету і вимагають подібних обчислень, ми ділимо рівні між цими функціями, так що $E_{\psi} = f_{\psi} \circ C_{\psi}$ і $T_{\psi,\omega} = D \circ g \circ (C_{\psi}, E_{\psi})$, де g - функція, яка поєднує вихідні дані кодера з нижчим шаром.

По-друге, враховуючи зацікавленність в максимальному збільшенні MI і не турбуючись його точним значенням, можна покластися на розбіжності, не пов'язані з KL, які можуть запропонувати вигідні компроміси. Наприклад, можна визначити оцінювач MI Дженсена-Шеннона:

$$\hat{L}_{\omega,\psi}^{(infoNCE)}(X; E_{\psi}(X)) := \mathbb{E}_{\mathbb{P}}\left[T_{\psi,\omega}(x, E_{\psi}(x)) - \mathbb{E}_{\tilde{\mathbb{P}}}\left[\log \sum_{x'} e^{T_{\psi,\omega}(x', E_{\psi}(x))}\right]\right], \quad (1.4)$$

де x – вхідний зразок,

x' – вхід, відібраний з $\tilde{\mathbb{P}} = \mathbb{P}$,

$sp(z) = \log(1 + e^z)$ – функція softplus.

Подібний оцінювач з'явився у контексті мінімізації сумарної кореляції, і це становить знайому двійкову перехресну ентропію. Це добре розуміється з точки зору оптимізації нейронних мереж, що на практиці такий підхід працює краще (наприклад, є стабільнішим), ніж ціль на основі DV. Інтуїтивно

оцінювач, що базується на Йенсена-Шеннона, повинен поводитись подібно до оцінювача, що базується на DV в рівнянні. 1.4, оскільки обидва діють як класифікатори, цілі яких максимізують очікуване *log*-співвідношення з'єднання над добутком маржиналів.

Оцінка контрастності шуму (Noise-Contrastive Estimation) також може використовуватися з DIM шляхом максимізації:

$$\hat{L}^{infoNCE}_{\omega,\psi}(X; E_\psi(X)) := \mathbb{E}_{\mathbb{P}}\left[T_{\psi,\omega}(x, E_\psi) - \mathbb{E}_{\tilde{\mathbb{P}}}\left[\log \sum_{x'} e^{T_{\psi,\omega}(x', E_\psi(x))}\right]\right]. \quad (1.5)$$

Для DIM ключовою різницею між формулюваннями DV, JSD та infoNCE є те, чи з'являється очікування над $\mathbb{P}/\tilde{\mathbb{P}}$ всередині *log* або поза ним. Насправді ціль, що базується на JSD, відображає оригінальну формулювання NCE, яка формулює ненормалізовану оцінку щільності як двійкову класифікацію між розподілом даних та розподілом шуму. DIM встановлює розподіл шуму на добуток граничних значень на X/Y, а розподіл даних − на справжнє з'єднання. Формулювання infoNCE слідує версії NCE на основі softmax, подібній до тих, що використовуються у спільноті моделювання мов і яка має міцні зв'язки з бінарною перехресною ентропією в контексті порівняльного навчання. На практиці реалізації ці оцінювачи здаються досить схожими і можуть повторно використовувати більшість того самого коду. Дослідження JSD та infoNCE у експериментах дозволяє виявити, що використання infoNCE часто перевершує JSD у подальших завданнях, хоча цей ефект зменшується із більш складними даними. Однак для того, щоб infoNCE та DV вимагали великої кількості негативних зразків (зразки з $\tilde{\mathbb{P}}$), вони повинні бути конкурентоспроможними.

Завдання в рівнянні 1.3 можна використовувати для максимізації MI між входом і виходом, але в кінцевому рахунку це може бути небажаним залежно від завдання. Наприклад, тривіальний шум на рівні пікселів марний для класифікації зображень, тому подання може не отримати вигоди від кодування цієї інформації (наприклад, при навчанні з нульовим знімком, навчанні передачі тощо). Для того, щоб отримати представлення, більш придатне для класифікації, можна замість цього максимізувати середній MI

між представленням високого рівня та локальними плямами зображення. Оскільки одному і тому ж представництву рекомендується мати високий MI з усіма патчами, це надає перевагу кодуванню аспектів даних, які спільно використовуються між виправленнями.

Припустимо, що вектор ознак має обмежену ємність (кількість одиниць та діапазон) і припустимо, що кодер не підтримує нескінченні вихідні конфігурації. Для максимізації MI між усім входом і поданням, кодер може вибрати який тип інформації на вході передається через кодер, наприклад, шум, характерний для локальних латок або пікселів. Однак, якщо кодер передає інформацію, характерну лише для деяких частин вводу, це не збільшує MI з будь-якими іншими патчами, які не містять згаданого шуму. Це спонукає кодер віддавати перевагу інформації, яка передається у вхідних даних.

DIM-фреймворк представлено на рис. 1.3.

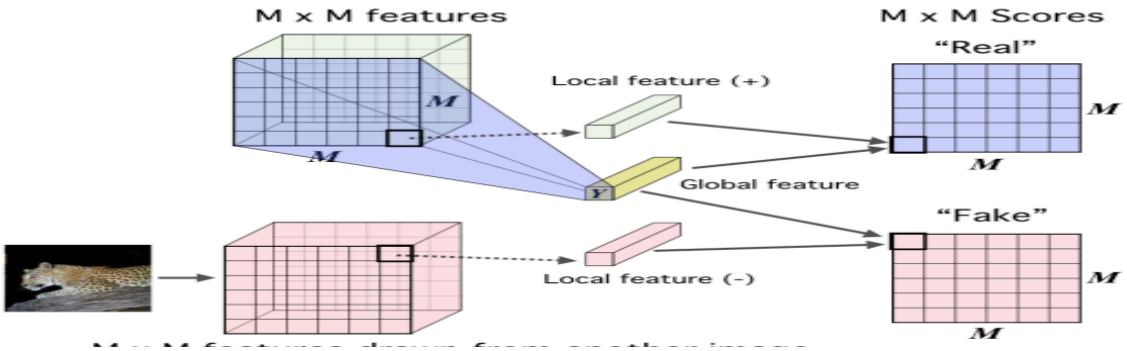

Рисунок 1.3 – Максимізація взаємної інформації між локальними та глобальними характеристиками.

Спочатку потрібнозакодувати вхідні дані до карти об'єктів, $C_\psi = \left\{C_\psi^{(i)}\right\}_{i=1}^{M\times M}$, що відображає корисну структуру в даних (наприклад, просторову місцевість), індексовану в цьому випадку i. Далі, можна узагальнити цю локальну карту об'єктів у загальну характеристику: $E_\psi = f_\psi \circ C_\psi(x)$. Потім необхідно визначити оцінювач MI на глобальних та локальних парах, максимізуючи середню оцінку MI:

$$(\hat{\omega}, \hat{\psi})_L = \arg\max_{\omega,\psi} \frac{1}{M^2} \sum_{i=1}^{M^2} \hat{L}_{\omega,\psi}(C_\psi^{(i)}(X); E_\psi(X)). \tag{1.6}$$

Абсолютна величина інформації є лише однією бажаною властивістю уявлення. Залежно від програми хороші уявлення можуть бути компактними, незалежними, розплутаними або незалежно керованими. DIM накладає статистичні обмеження на вивчені уявлення, неявно навчаючи кодер так, щоб образ міри $\mathbb{U}_{\psi,\mathbb{P}}$ збігався з попереднім $\mathbb{V}$. Це робиться шляхом навчання дискримінатора, $D_\phi : Y \to \mathbb{R}$, для оцінки розбіжності, $D(\mathbb{V}||\mathbb{U}_{\psi,\mathbb{P}})$. Після чого тренуємо кодер для мінімізації оцінки:

$$\begin{aligned}(\hat{\omega}, \hat{\psi})_P \arg\min_{\psi} \arg\max_{\phi} \hat{D}_\phi(\mathbb{V}||\mathbb{U}_{\psi,\mathbb{P}}) \\ = \mathbb{E}_{\mathbb{V}}[\log D_\phi(y)] + \mathbb{E}_{\mathbb{P}}[\log(1 - D_\phi(E_\psi(x)))].\end{aligned} \tag{1.7}$$

Цей підхід подібний до того, що робиться в змагальних автокодерах, але без генератора. Він також схожий на шум як цілі, але тренує кодер, щоб він неявно відповідав шуму, а не використовував апріорні вибірки шуму як цілі.

Всі три цілі: глобальна та локальна максимізація MI та попереднє узгодження можуть бути використані разом. І, таким чином, можна досягнути своєї повної мети для Deep InfoMax (DIM):

$$\begin{aligned}\arg\max_{\omega_1,\omega_2,\psi}(\alpha \hat{L}_{\omega_1,\psi}(X; E_\psi(X)) + \frac{\beta}{M^2}\sum_{i=1}^{M^2} \hat{L}_{\omega_2,\psi}(X^{(i)}; E_\psi(X))) \\ + \arg\min_{\psi} \arg\max_{\phi} \gamma \hat{D}_\phi(\mathbb{V}||\mathbb{U}_{\psi,\mathbb{P}}),\end{aligned} \tag{1.8}$$

де ω_1 та ω_2 – параметри дискримінатора відповідно до глобальних та локальних цілей, а α, β та γ – гіперпараметри.

1.3 Momentum Contrast

Порівняльне навчання та його останні розробки можна розглядати як навчання кодеру для завдання пошуку словника, як описано далі.

Розглянемо закодований запит q та набір закодованих зразків $\{k_0, k_1, k_2, \dots\}$, які є ключами словника. Припустимо, що у словнику є один ключ (позначений як k_+), який відповідає q. Контрастивна втрата – це функція, значення якої є низьким, коли q подібне до його позитивного ключа

k_+ та відрізняється від усіх інших ключів (вважається негативним ключем для q). Можна перевизначити функцію InfoNCE:

$$L_q = -\log \frac{exp(q \cdot k_+/\tau)}{\sum_{i=0}^{K} exp(q \cdot k_i/\tau)}, \tag{1.9}$$

де τ гіперпараметр. Сума складається з однієї позитивної та K негативних екземплярів. Інтуїтивно, ця втрата є часовою втратою класифікатора на основі $(K + 1)$ softmax, який намагається класифікувати q як k_+. Функції контрастивних збитків можуть також базуватися на інших формах, таких як маржові збитки та варіанти збитків NSE.

Контрастивна втрата служить неконтрольованою цільовою функцією для навчання мереж кодера, що представляють запити та ключі. Загалом, подання запиту має значення $q = f_q(x^q)$, де f_q - це мережа кодера, а x^q - зразок запиту (так само, $k = f_k(x^k)$). Їх екземпляри залежать від конкретного попередньогозавдання. Вхідними даними x^q та x^k можуть бути зображення, патчі або контекст, що складається з набору патчів. Мережі f_q і f_k можуть бути ідентичними, частково спільними або різними.

З вищенаведеної точки зору, порівняльне навчання − це спосіб побудови дискретного словника на безперервних вхідних даних, таких як зображення. Словник є динамічним у тому сенсі, що ключі вибираються випадково, а кодер ключів розвивається під час навчання. Гіпотеза полягає в тому, що хорошим можливостям можна навчитися за допомогою великого словника, який охоплює багатий набір негативних зразків, тоді як кодер ключів словника зберігається якомога послідовнішим, незважаючи на його розвиток. Виходячи з цього можна описати алгорим Momentum Contrast.

В основі цього підходу лежить підтримка словника як черги зразків даних. Це дозволяє нам повторно використовувати закодовані ключі з безпосередніх попередніх міні-пакетів. Введення черги відокремлює розмір словника від розміру міні-партії. Розмір словника може бути набагато більшим, ніж типовий розмір міні-партії, і може бути гнучко та незалежно встановлений як гіперпараметр.

Зразки у словнику поступово замінюються. Поточна міні-партія потрапляє до словника, а найстаріша міні-партія в черзі вилучається. Словник завжди представляє вибіркову підмножину всіх даних, тоді як додаткові обчислення ведення цього словника є керованими. Більше того, видалення найстарішої міні-партії може бути корисним, оскільки її закодовані ключі є найбільш застарілими і, отже, найменш відповідають найновішим.

Використання черги може зробити словник великим, але це також робить важким оновлення кодера ключа шляхом зворотного розповсюдження (градієнт повинен поширюватися на всі зразки в черзі). Наївним рішенням є копіювання кодера ключа f_k з кодера запиту f_q, ігноруючи цей градієнт. Але це рішення дає погані результати в експериментах. Припустимо, що такий збій спричинений швидко мінливим кодером, який зменшує узгодженість ключових подань. Пропонується оновити імпульс для вирішення цієї проблеми.

Формально, позначаючи параметри f_k як θ_k, а параметри f_q як θ_q, ми оновлюємо θ_k за допомогою:

$$\theta_k \leftarrow m\theta_k + (1-m)\theta_q. \tag{1.10}$$

Тут $m \in [0, 1)$ – коефіцієнт імпульсу. Тільки параметри θ_q оновлюються шляхом зворотного розповсюдження. Оновлення імпульсу в рівнянні (6) змушує θ_k еволюціонувати більш плавно, ніж θ_q. Як результат, хоча ключі в черзі кодуються різними кодерами (в різних міні-партіях), різниця між цими кодерами може бути невеликою. В експериментах порівняно великий імпульс (наприклад, m = 0,999, за замовчуванням) працює набагато краще, ніж менший показник (наприклад, m = 0,9), що дозволяє припустити, що повільно розвивається кодер ключа є стрижнем для використання черги.

MoCo – загальний механізм використання контрастивних втрат. Ми порівнюємо це з двома існуючими загальними механізмами на рис 1.4. Вони мають різні властивості щодо розміру та послідовності словника.

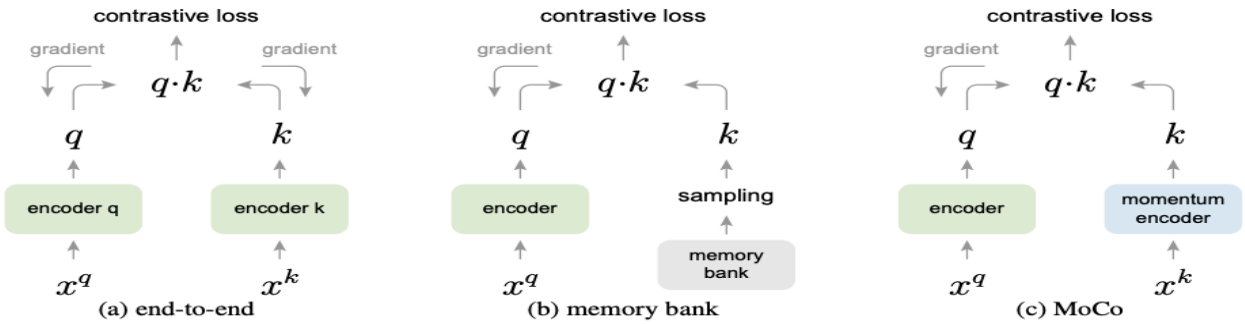

Рисунок 1.4 – Концептуальне порівняння трьох контрастивних механізмів втрат.

Оновлення end-to-end шляхом зворотного розповсюдження є природним механізмом. Він використовує зразки в поточній міні-партії як словник, тому ключі послідовно кодуються (тим самим набором параметрів кодера). Але розмір словника поєднується з міні-пакетним розміром, обмеженим обсягом пам'яті графічного процесора. Це також кидається виклик великій міні-пакетній оптимізації. Деякі останні методи засновані на завданнях підтексту, керованих місцевими позиціями, де розмір словника може бути збільшений на кілька позицій. Але для цих завдань-претекстів можуть знадобитися спеціальні мережеві конструкції, такі як патчіфікація вводу або налаштування сприйнятливого розміру поля, що може ускладнити перенесення цих мереж на подальші завдання

Іншим механізмом є підхід банк пам'яті (memory bank). Банк пам'яті складається з подань усіх зразків у наборі даних. Словник для кожної міні-партії вибірково вибирається з банку пам'яті без зворотного розповсюдження, тому він може підтримувати великий розмір словника. Однак представлення вибірки в банку пам'яті було оновлено, коли її востаннє бачили, тому вибіркові ключі, по суті, стосуються кодерів на декількох різних етапах протягом усієї минулої епохи і, отже, менш послідовні. Оновлення імпульсу приймається на банку пам'яті. Її імпульс оновлення подається на поданнях того самого зразка, а не на кодері. Це оновлення імпульсу не має значення для нашого методу, оскільки MoCo не відстежує кожен зразок. Більше того, наш метод є більш ефективним в пам'яті, і його можна навчити на мільярдних даних, що може бути важко для банку пам'яті.

Порівняльне навчання може працювати з різними попередніми задачами.

Можна розглянути запит і ключ як позитивну пару, якщо вони походять з одного зображення, а в іншому — як негативну пару вибірки. Візьмемо два випадкові «види» одного і того ж зображення під час випадкового збільшення даних, щоб сформувати позитивну пару. Запити та ключі кодуються відповідними кодерами f_q та f_k. Кодером може бути будь-яка згорткова нейронна мережа.

Алгоритм

Momentum Contrast для попередньої задачі можна реалізовувати по-різному. Наприклад: адаптувати нейронну мережу ResNet як кодер, чий останній повністю підключений шар (після загального середнього об'єднання) має вихід з фіксованим розміром (128-D). Цей вихідний вектор нормується за його L2-нормою. Це представлення запиту або ключа. Після чого потрібно обрати значення для аргументу τ з 1.9. Далі наведено параметр збільшення даних: обрізання 224×224 пікселів береться із зображення випадкового розміру, а потім піддається випадковому коливанню кольорів, випадковому горизонтальному перевертанню та випадковому перетворенню шкал сірого.

Обидва кодери f_q і f_k використовують пакетну нормалізацію (Batch Normalization, BN), як у стандартному ResNet. Можна показати, що використання BN заважає моделі засвоїти хороші уявлення.

Цю проблему можливо вирішити за допомогою тасування BN. Можна тренувати модель з декількома графічними процесорами та виконуввати BN на зразках незалежно для кожного графічного процесора (як це робиться у звичайній практиці). Для кодера ключів f_k перемішується порядок зразків у поточній міні-партії перед розподілом між графічними процесорами (і перемішується назад після кодування). Порядок вибірки міні-партії для кодера запиту f_q не змінюється. Це гарантує, що статистична інформація про партії, яка використовується для обчислення запиту, та її позитивний ключ надходять із двох різних підмножин. Це ефективно вирішує проблему дозволяє тренуванню принести користь від BN.

Використання перемішаного BN може стати у нагоді також у end-to-end оновленні. Це не має значення для банку пам'яті, який не страждає від цієї

проблеми, оскільки позитивні ключі були від різних міні-пакетів у минулому.

1.4 Висновки

Самонавчання дає нам змогу безкоштовно використовувати різноманітні позначки, що постачаються з даними. Мотивація досить проста: створення набору даних із чіткими мітками є дорогим, але постійно створюються немічені дані. Щоб використовувати набагато більшу кількість немаркованих даних, одним із способів є правильне встановлення цілей навчання, щоб отримати контроль від самих даних.

Порівняльне навчання - це техніка машинного навчання, яка використовується для вивчення загальних особливостей набору даних без міток, навчаючи моделі, які точки даних подібні чи різні.

У данній роботі використовуються алгоритми Deep InfoMax (DIM) та Momentum Contrast.