Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мережних технологій факультету інформатики

ПОБУДОВА СЕМАНТИЧНОЇ МОДЕЛІ ЗОБРАЖЕННЯ

Текстова частина до курсової роботи
за спеціальністю „Комп'ютерні науки" 122

Керівник курсової роботи
д-р. т.н., доц. Глибовець А.М.
_____
(підпис)
"____" _____ 2021 р.

Виконала студентка КН-4
Лопухіна С.В.
"____" _____ 2021 р.

Київ 2021

Календарний план

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітки
1.	Отримання завдання на курсову роботу.	23.10.2020	
2.	Огляд попередніх напрацювань за темою роботи.	20.11.2020	
3.	Аналіз останніх робіт з цієї області	31.01.2021	
4.	Початок розробки власної моделі	01.03.2021	
5.	Написання текстової частини роботи	30.03.2021	
6.	Створення презентації	10.04.2021	
7.	Подання роботи на кафедру для перевірки на плагіат	12.04.2021	
8.	Захист курсової роботи	19.04.2021	

Студент _____

Керівник _____

“_____” _____

Зміст

Анотація

У даній роботі аналізуються роботи з області машинного навчання, що вирішують задачу знаходження візуальних зв'язків на зображені, або іншими словами, будують семантичні моделі зображень. На основі аналізу цих робіт робиться спроба побудувати власну модель для побудови семантичної моделі зображення, використовуючи підхід Transfer Learning.

Вступ

Метою цієї роботи є побудова семантичної моделі зображення, для того щоб наблизитися до вирішення проблеми розуміння зображення комп'ютером. Ця проблема є актуальною та має не так багато рішень, як інші задачі комп'ютерного зору, такі як класифікація зображень або створення текстового опису зображення.

У першому розділі цієї роботи описується актуальність та розглядається проблематика задачі побудови семантичної моделі зображення. У другому розділі розглянуто та проаналізовано існуючі рішення для задач класифікації зображень, текстового опису зображень та знаходження візуальних зв'язків на зображеннях. У третьому розділі описано процес побудову власної моделі та результати її роботи.

Розділ 1. Семантична модель зображення

1.1 Розуміння зображення комп’ютером

Більшу частину інформації навколо себе ми сприймаємо завдяки нашим очам. І найшвидше ми сприймаємо інформацію саме у вигляді зображень – нейробіологи з Массачусетського технологічного інституту виявили, що мозок може розпізнавати зображення всього за 13 мілісекунд [1]. Отже, у сучасному світі, переповненому інформацією, зображення представляють собою один з найбільш зручних способів для обміну інформацією.

Нам відомо, що інформація накопичується доволі швидко й з кожною секундою її стає більше [2]. Для пошуку потрібної нам інформації, її фільтрування та сортування ми користуємося комп’ютерами та їх можливостями. Комп’ютери все ще не здатні розуміти зображення так само швидко й у тій же повній мірі, що і людський мозок – і через це комп'ютерам складніше фільтрувати та сортувати зображення. Оскільки зображення становлять значну частку всієї інформації у Інтернеті, питання розуміння комп’ютерами зображення стає з кожним днем все актуальнішим. Розглянувши як саме розуміє зображення людина та як це робить комп’ютер можна зрозуміти, у якому напрямку нам слід рухатися для вирішення цього питання.

Люди та комп’ютери поки що сприймають зображення по різному. Людина при перегляді зображення інтерпретує його виходячи з свого досвіду — вона розуміє що зображено завдяки попередньо накопиченим знанням про різні об’єкти з реального світу та їх взаємозв’язки. Більш того, люди здатні зрозуміти зображений об’єкт навіть якщо він розмитий або його не видно повністю, оскільки ми здатні за допомогою уяви додумувати потрібні елементи. Людина може легко визначити, що саме зображено, що відбувається на зображенні, чи елементи на ньому пов’язані між собою та як саме і в результаті після цього на основі власного досвіду надати цьому всьому якусь інтерпретацію.

Ми здатні так швидко інтерпретувати що знаходиться на зображенні в основному завдяки своєму досвіду – наш мозок за всі роки нашого життя накопичує багато прикладів як виглядає той або інший об’єкт, які у об’єктів можуть бути взаємозв’язки та як вони зазвичай виглядають, тощо.

У той же час, комп’ютерна інтерпретація зображень набагато примітивніша. Для комп’ютера зображення – це набір чисел, кожне з яких є представленням пікселя зображення. І це за умови що зображення чорно-біле, а якщо воно кольорове то його представлення набуває вигляду трьох наборів чисел, кожен з яких відповідає за один з каналів RGB. Таке представлення кольорового зображення тільки ускладнює його сприйняття комп’ютером – навіть задача пошуку простих геометричних фігур стає нетривіальною, вже не кажучи про довільні об’єкти.

Отже через те, як швидко збільшується кількість даних у світі, в тому числі зображень, та через потребу у їх аналізі та фільтрації врешті решт виникла окрема галузь комп’ютерних наук – комп’ютерний зір. Ця галузь поставила собі на меті автоматизувати обробку візуальної інформації та наблизитися до того, як це робить людський мозок. Наразі методи комп’ютерного зору досягли значного прогресу в вирішенні деяких задач обробки зображень – класифікація, визначення меж об’єктів, сегментація та інші. Це, звісно, є значним покращенням відносно простого сприйняття комп’ютером зображення як масиву пікселів, але все це не дуже сильно наближає нас до рівня розуміння людини. Але з іншого боку, ці розробки є важливою та потужною базою для подальших досліджень і покращень результатів.

У обробці зображень однією з складнощів є розробка алгоритму для вирішення певної задачі. Як приклад можна навести класичну задачу класифікації зображень – а саме знаходження певних класів з наперед визначеного набору на зображені. Простіше це можна було б описати як пошук відповіді на питання “що зображено на картинці?”, при тому що відомі варіанти відповіді. Найпростішим випадком цієї задачі є бінарна класифікація

– коли просто потрібно вказати, чи зображення належить до певного класу, чи ні. Якщо, наприклад, ми маємо клас “яблуко”, то виникає питання: як визначити, що на зображенні воно є? Який алгоритм ми використаємо для цього?

Людина б сказала що відповідь доволі очевидна: яблуко – шароподібний об’єкт, може мати зелений, червоний або жовтий колір, росте на дереві. Кожен бачив у своєму житті яблуко і має точне розуміння, як воно виглядає. Разом з тим, нам не потрібно знати як виглядає будь-який сорт яблук на цій планеті для того, щоб дати відповідь на питання “як виглядає яблуко?”. Для людини алгоритм визначення “чи це яблуко” доволі простий й базується на вже наявних знаннях про яблука та їх звичайний вигляд.

Можна спробувати побудувати схожий алгоритм для комп’ютера – він буде перевіряти зображення на наявність характерних риси яблук для того щоб визначити, чи присутні вони на зображені. Але одразу ж виникає кілька питань: наскільки багато має бути цих рис, які з них важливіші, що робити, якщо знайдено тільки частину рис, а не всі? Існує багато інших фруктів та овочів, які схожі на яблука, тому треба розуміти як розрізнити чи це помідор чи яблуко, або загалом апельсин. Крім того, на зображені навряд чи буде тільки одне яблуко на білому фоні – там можуть бути декілька яблук, вони можуть бути тільки в куточку, саме зображення може бути сильно деформоване – і це все що тільки ще більше ускладнює нашу задачу.

Очевидним стає одне: створити такий алгоритм або неможливо, або дуже складно. До того ж що робити, якщо ми захочемо потім все таки навчити комп’ютер класифікувати ще й помідори? Для кожного нового класу потрібно буде створювати новий алгоритм, підбирати нові риси, тощо. Враховуючи все це, можемо зробити висновок що підхід з явним визначенням алгоритму не найкращий і потрібно шукати інший підхід до цієї задачі.

Таким підходом є машинне навчання. Замість явного задання алгоритму, машинне навчання створює свій власний на основі великої кількості вхідних даних. Такий підхід імітує те, як люди здобувають досвід і завдяки цьому

навчаються вирішувати різні задачі. Цей спосіб надає нам можливість не явно задавати алгоритм, а просто надати багато прикладів на вхід, для того щоб машина сама розробила алгоритм для розрізнення цих прикладів (якщо мова йде про класифікацію).

Оскільки машинне навчання добре підходить для вирішення задач з зображеннями, його почали активно використовувати в обробці зображень. Завдяки ньому, було вдало розв’язано багато нових задач, які раніше не можна було вирішити. Але у машинного навчання є свої складнощі – для такого підходу необхідна велика кількість відповідних вхідних даних. Нові дані, у тому числі зображення, як було згадано раніше, з’являються у великих кількостях кожного дня. З одного боку, це спрощує задачу – немає бути проблеми з тим, щоб знайти потрібні приклади для певної задачі. З іншого боку, їх кількість ускладнює цей пошук та задачу створення повного набору рис для вхідних даних.

Але, незважаючи на складнощі, машинне навчання є наразі найкращим підходом, який допоможе нам у вирішенні питання розуміння комп’ютером зображень і наближенням вирішення цієї задачі до рівня людини.

1.2 Практичне застосування розуміння зображень

Перш ніж перейти до опису рішення щодо розумінь зображення, скажемо декілька слів щодо корисності та використання цього. Як саме інтерпретація зображення комп’ютером може допомогти людям, які самостійно чудово вирішують це питання?

Насправді, існує доволі багато прикладів застосувань комп’ютерного зору. Знову згадуючи великі об'єми даних, ми розуміємо що для їх фільтрування та сортування було б доречно використовувати саме комп’ютерний зір. За допомогою тієї самої класифікації можно значно спростити пошук по картинках, оскільки за умови що комп’ютер буде сам розуміти що знаходиться на зображені відпаде потреба у достовірних мета-тегах та описах зображень. Не завжди у всіх вистачає наснаги детально

описати все, що знаходиться на зображенні, через що пошуковики, наприклад, не в змозі повноцінно індексувати всі зображення та видавати потрібні картинки на певні запити.

Комп'ютерний зір має застосування й у інших галузях пов’язаних з обробкою зображень – автоматична обробка, видалення або заміна фону, видалення елементів або автоматичне доповнення пошкоджених або неякісних зображень, тощо.

Зважаючи на те, що зображень стає дедалі більше то всі ці задачі вже неможливо виконувати тільки за рахунок ручної праці. Так, для людей виконання цих задач є набагато простішим порівняно з комп’ютерами. Але роботу людей важко масштабувати і, до того ж, вони мають обмежену швидкість. Внаслідок цього можна стверджувати, що автоматизація обробки зображень — це необхідність у сьогоднішньому світі.

Тепер, коли ми розуміємо необхідність автоматичної обробки та варіанти застосування розуміння зображень комп’ютером, варто поставити питання: яка задача є найбільш фундаментальною в обробці зображень? Що допомогло б нам узагальнити і вирішити проблему розуміння зображень? Вищезгадана задача класифікації зображень, або у більш загальному випадку, задачу пошуку об’єктів на зображенні, дає нам відповідь тільки на питання “що зображено?”. Але людина розуміє зображення у більшій мірі — і такі речі як просторове розташування, зв’язки об'єктів на зображені та інші не враховуються у задачі класифікації. Тому ця задача є тільки першим наближенням до вирішення задачі розуміння зображень.

Мета даної роботи — запропонувати краще наближення, яке буде більш точно описувати та розуміти зображення.

1.3 Семантична модель зображення

Для цього спочатку введемо поняття “семантична модель” — формалізоване представлення зображеного на певній картинці. “Семантична” означає, що це представлення має розкривати семантику зображеного, тобто

повинне бути максимально наближеним до того, як зображення інтерпретуються людьми. “Модель” означає, що представлення має бути формалізоване, а саме мати доволі чітко визначену структуру.

Зауважимо, що чітка формальна структура є дуже важливою, тому що ми говоримо про автоматизовані системи. Якщо наші дані будуть неструктуровані або нечіткі, то це доволі сильно ускладнить нам їх використання або й зовсім унеможливить.

Тепер, коли ми формалізували поняття семантичної моделі, можна починати розробку рішення питання розуміння зображення комп’ютером та наближення цього рішення до рівня людини. Але, звичайно, вже існують підходи, метою яких було вирішити цю задачу. Тому перед тим як будувати своє власне рішення, ми розглянемо та проаналізуємо вже існуючі, щоб оцінити їх недоліки та переваги та обрати найбільш досконалі на даний момент.

Найбільш досконалими будемо вважати ті рішення, що будуть найбільш наближені до точної імітації людського розуміння зображення комп’ютером. Для цього комп’ютеру потрібно було б набути тих самих навичок для аналізу зображення що й людині, і такі підходи ми й будемо шукати.

Розділ 2. Аналіз існуючих рішень для побудови семантичної моделі зображення

2.1 Підзадачі вирішення нашої проблеми

Перш ніж обрати які підходи ми будемо використовувати для побудови рішень проблеми розуміння зображення, розглянемо на які підзадачі розбивається дана проблема. Оскільки у галузі комп'ютерного зору існує досить багато інших проблем, пов'язаних з нашою, варто розглянути ці підходи для кращого розуміння як саме ми можемо частково вирішити нашу проблему з їх допомогою. Маючи уявлення про те, які підзадачі перед нами стоять, ми зможемо правильно спланувати подальші кроки.

Для того, щоб мати можливість побудувати семантичну модель зображення, а саме визначати що на ньому знаходиться та як ці об'єкти пов'язані нам буде потрібно вирішити такі підзадачі: класифікація зображень, текстовий опис зображень та знаходження зв'язків між об'єктами на зображені.

Отже надалі буде розглянуто ці задачі комп'ютерного зору і для кожної ми сформулюємо: в чому її суть, як вона вирішується і як саме це допоможе нам у проблемі побудови семантичної моделі.

2.2 Класифікація зображень

Класифікація зображень є фундаментальним завданням, яке вирішується комп'ютерним зором. Класифікація намагається зрозуміти усе зображення як одне ціле. Його мета - класифікувати зображення, присвоївши йому певну мітку. Зазвичай класифікація зображень займається зображеннями, на яких з'являється та аналізується лише один об'єкт. На відміну від цього, виявлення об'єктів включає як класифікацію, так і завдання локалізації і використовується для аналізу більш реалістичних випадків, коли на зображенні може існувати кілька об'єктів.

Для нашого випадку звичайна класифікація не підійде, оскільки кожна людина здатна зрозуміти що на зображені знаходиться декілька об’єктів та ідентифікувати їх всіх. Зважаючи на це і на те, що ми будемо намагатися наблизити комп’ютерне розуміння зображення до людського, ми звернемо увагу саме на виявлення об’єктів та багатокласову класифікацію.

Багатокласова класифікація у машинному навчанні — це проблема класифікації екземплярів даних (у нашому випадку зображень) в один із трьох або більше класів. Слід зауважити, що багатокласову класифікацію не слід плутати з класифікацією з багатьма помітками, де для кожного екземпляра у результаті передбачається наявність кількох міток.

Найкраще з вирішення задачі класифікації наразі показують себе нейронні мережі, зокрема глибокі нейронні мережі. Саме їх приклади та досягнення у вирішенні цієї задачі буде розглянуто надалі.

Одним з найбільш популярних наборів даних для класифікації зображень є ImageNet. На основі даних з сайту PaperWithCode [3] ми розглянемо, які моделі на даних з датасету ImageNet показували себе найкраще та зробимо висновки які з них найкраще підійдуть для вирішення нашої задачі. Надалі буде надано короткий опис для чотирьох найкращих попередньо навчених моделей для класифікації зображень, які є досі вважаються state-of-the-art моделями та широко використовуються. Також буде розглянуто модель з найкращими показниками станом на квітень 2021 року, для розуміння того який є рівень сучасних досягнень у вирішенні задачі класифікації зображень.

2.2.1 Very Deep Convolutional Networks for Large-Scale Image Recognition(VGG-16)

VGG-16 - одна з найпопулярніших попередньо навчених моделей для класифікації зображень. Ця модель була і залишається тою, яку складно перемогти навіть сьогодні. Вона була розроблена в Оксфордському університеті, і перевершила тодішній стандарт AlexNet, через що її швидко почали глобально використовувати для завдань класифікації зображень.

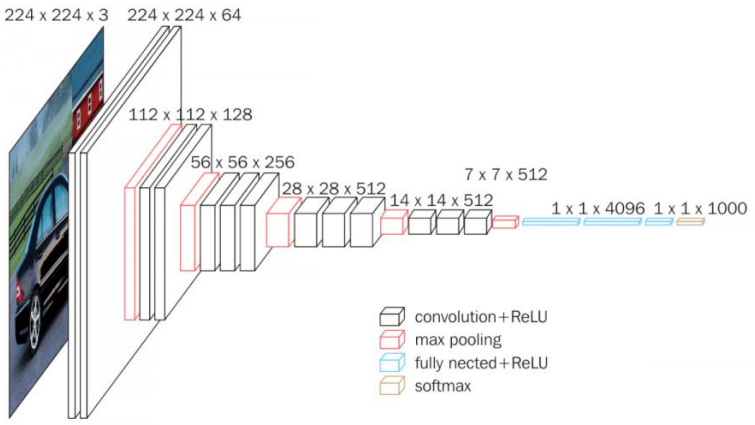


Рис 2.1. Архітектура моделі VGG-16

Ця модель має послідовний характер і використовує безліч фільтрів. Але її недоліком є її загальна кількість параметрів, що становить 138 млрд., і це робить модель доволі повільною. Якщо мова йде про великі набори даних, то ця модель є не найкращим варіантом у порівнянні з іншими.

Існує вдосконалений варіант моделі VGG16, наприклад VGG19 (містить 19 шарів). Ця модель навчена на більшій кількості зображень і краще вирішує цю проблему, але все ще є так само як і її попередник доволі великою та повільною.

2.2.2. Inception

У 2014 році було створено не тільки популярну VGG-16 — Inception було створено того ж року і якщо VGG-16 отримав 2-е місце у ILSVRC того року, 1-е місце зайняла ця модель розроблена Google.

Спочатку це була модель Inception v1, що мала лише 7 мільйонів параметрів (це було набагато менше, ніж у поширених на той час VGG та AlexNet). Потім була розроблена наступна вдосконалена версія цієї моделі — Inceptionv2. У ній було підвищено точність і сама модель була спрощена. У тій же роботі, що і Inception v2, автори представили модель Inception v3 з ще кількома удосконаленнями до v2.

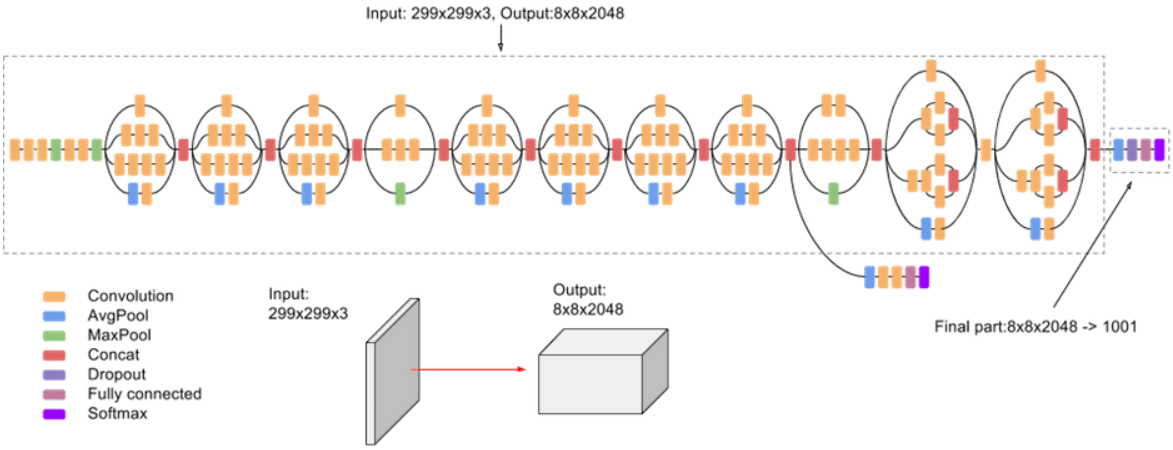


Рис. 2.2. Архітектура моделі Inception v3 [5]

Основними вдосконаленнями у Inception v2 були впровадження нормалізації батчів, збільшення факторизації та додавання оптимізатора RMSProp. В результаті Inception v3 досягла найвищої позиції в CVPR 2016 з показником top-5 error rate у 3,5%.

2.2.3 ResNet50

Оригінальна модель називалася Residual net або ResNet і стала ще одним новим етапом у сфері комп’ютерного зору ще в 2015 році.

Основною метою цієї моделі було уникнути низької точності, оскільки модель становилася глибшою у порівнянні з своїми сучасниками. Крім того, модель ResNet мала на меті вирішити проблему так званого зникаючого градієнту.

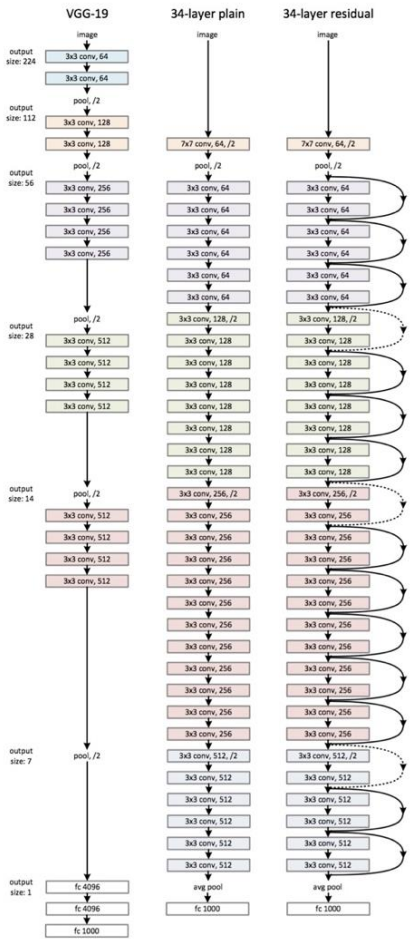


Рис 2.3. Архітектура ResNet34 (ResNet50 також побудований за схожою технікою, лише містить більшу кількість шарів) [6]

Основна концепція моделей ResNet полягає у пропущених з'єднаннях, які називаються "з'єднання скорочення ідентичності" — після кожних 2 слоїв у цій моделі ми пропускаємо шар між ними. Потім ці пропущенні з’єднання використовують так звані залишкові блоки:

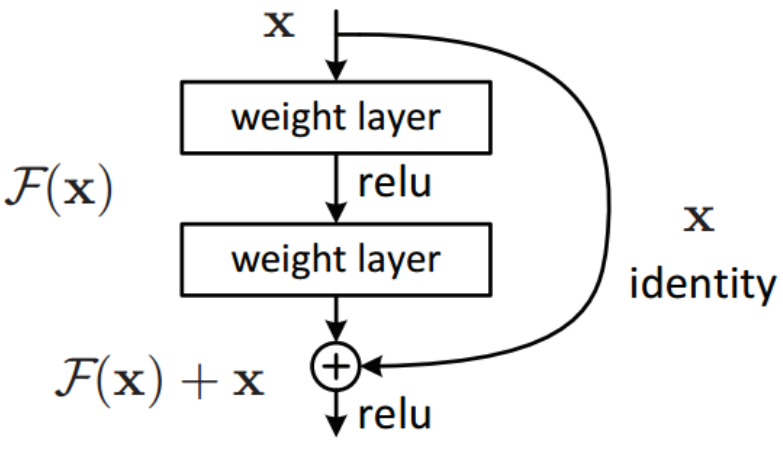


Рис 2.4. “Residual” блок моделі ResNet

Простіше кажучи, автори ResNet вважають, що підбір залишкового відображення набагато простіше, ніж встановлення фактичного, і через це вони застосовують його у всіх шарах. Цікавим є ще те, що автори ResNet дотримуються думки, що більша кількість шарів не спричиняє грішу роботу моделі, як це зазвичай вважається.

Модель ResNet має безліч варіантів, найновішим з яких є ResNet152. Далі подано архітектуру сімейства ResNet з точки зору використовуваних шарів:

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

ures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of block

Рис 2.5. Архітектура сімейства ResNet [8]

Крім того що ResNet50 є однією з найбільш широко використовуваних попередньо навчених моделей, вона породила серію архітектур, заснованих на ній. До цього переліку можна додати такі моделі як ResNeXt та FixResNeXt, які за останні роки обидві займали деякий час перше місце у рейтингу найкращих класифікаторів по ImageNet [3]. До цього всього можна ще тільки додати що ResNet50 є однією з найпопулярніших моделей, що має показник top-5 error rate близько 5%.

2.2.4. EfficientNet

Останньою моделлю, яку ми розглянемо, є ще одна модель від Google. У EfficientNet автори пропонують новий метод масштабування, який називається “складеним масштабуванням”. Більш ранні моделі, наприклад такі

як ResNet, дотримуються загальноприйнятого підходу до масштабування розмірів мережі — вони просто додають при масштабуванні все більше і більше шарів до мережі.

У цій же роботі пропонується підхід, при якому ми одночасно масштабуємо розміри на фіксовану величину і робимо це рівномірно — через це можна досягнути набагато кращих показників. До того ж, масштабні коефіцієнти визначаються користувачем.

Хоча цей прийом масштабування може бути використаний для будь-якої моделі на базі CNN, автори почали з власної базової моделі під назвою EfficientNetB0:

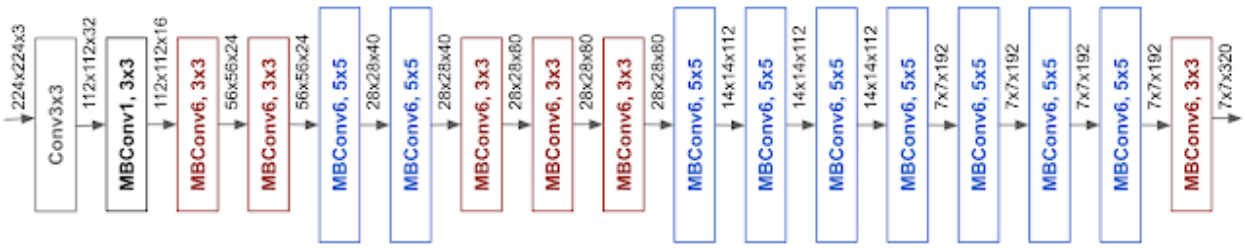


Рис. 2.6. Архітектура EfficientNetB0 [9]

Шар MBConv розшифровується як mobile inverted bottleneck Convolution. Розробники цієї моделі також пропонують підставляти у формулу складеного масштабування наступні коефіцієнти масштабування: глибина = 1,20, ширина = 1,10, роздільна здатність = 1,15.

Ці параметри використовуються для побудови сімейства EfficientNets - EfficientNetB0 до EfficientNetB7. Далі наведено графік, що демонструє порівняльну ефективність цього сімейства щодо інших популярних моделей:

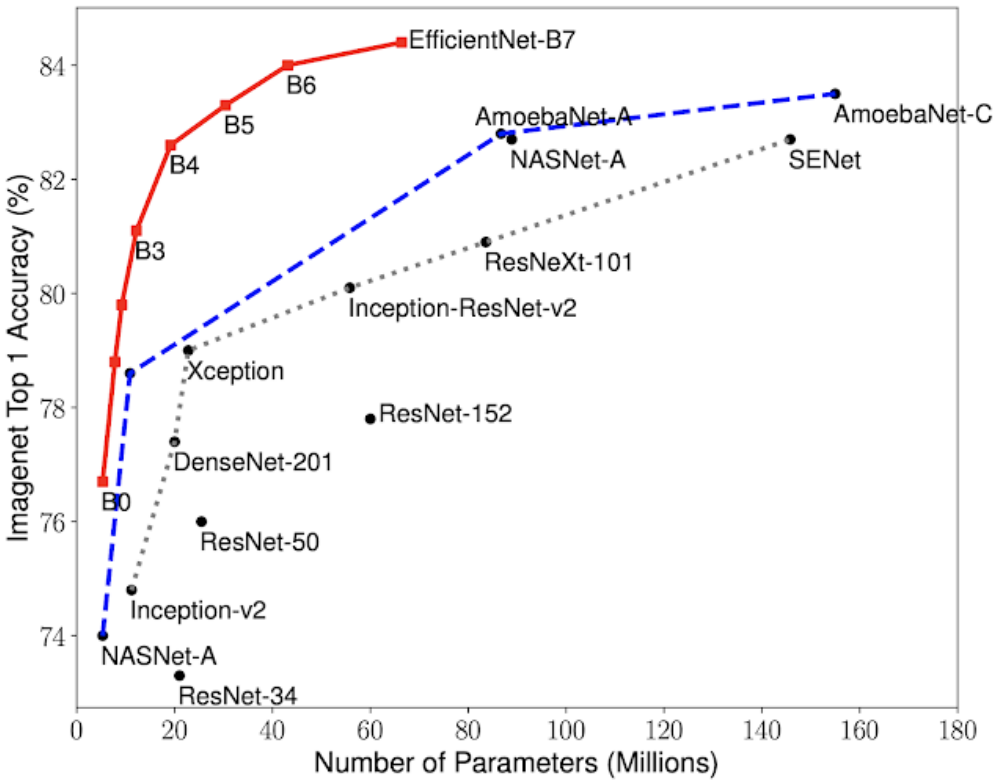


Рис. 2.7. Порівняння ефективності моделей сімейства EfficientNet з іншими популярними моделями

Як можна побачити, навіть базова модель B0 починається з набагато вищою точністю, яка лише продовжує зростати, і це все теж з меншою кількістю параметрів ніж інші моделі. Наприклад, EfficientB0 має лише 5,3 мільйона параметрів, що у порівнянні з 23 мільйонами ResNet становить значну різницю.

2.2.5. Meta Pseudo Labels

Станом на квітень 2021 року, відповідно до даних на сайті Paperswithcode [3], найкраще з завдання класифікації зображень показує себе модель Meta Pseudo Labels, з топ-1 точністю у 90.2%. Ця модель так само як і деякі з попередніх була розроблена Google і за її основу взято дві моделі EfficientNet-L2. Ця модель використовує напівконтрольований метод навчання, який досягає нової першокласної точності на 1,6% краще, ніж попередні моделі. У цій моделі є мережа викладачів, яка генерує псевдо мітки для немаркованих даних щоб навчити по ним учнівські мережі моделі. Викладач у Meta Pseudo Labels постійно адаптується за допомогою зворотного

зв’язку результатів роботи учня на маркованому наборі даних і в результаті він генерує кращі псевдо ярлики для навчання студента. Цей підхід є новим і завдяки ньому і було досягнуто найкращих на сьогоднішній день показників.

Для вирішення нашої задачі задача класифікації зображень становить собою одну з основних підзадач. З її допомогою ми зможемо правильно визначити всі об’єкти на зображені для подальшого їх опрацювання. Тому, незважаючи, на недостатність передбачення тільки одного класу для вирішення задачі побудови семантичної моделі, деякі ідеї з вищезгаданих праць з цієї області будуть адаптовані та використані у нашому рішенні для досягнення найкращих результатів.

2.3 Підписи до зображень

Суть задачі підписування зображень (image captioning) полягає в генеруванні текстового опису, зазвичай, у вигляді одного речення для вхідного зображення. Для наочного прикладу наводимо рис. 2.8.

Рис. 2.8. Приклади результатів вирішення задачі підписування зображень [10]

Вирішення цієї задачі допоможе нам у вирішенні проблеми розуміння семантики зображення — такий опис зображення хоч і не є формалізованим, але значно багатше описує зображення ніж вищеописана класифікація. За

допомогою текстового опису зображень ми маємо змогу не обмежуватися одним об’єктом на зображенні, а намагатися захопити ще й те, як він взаємодіє з іншими об’єктами, які у цих об’єктів можуть бути атрибути чи зв’язки, як саме вони пов’язані з їх оточенням, тощо.

Але варто зауважити, що хоча такий опис зображень є більш привабливим рішенням для нашої задачі, проте результати цього рішення є доволі суб’єктивними. Через те, що немає вимог до того, яким чином має бути структурований опис та чи має він включати всі об’єкти на зображенні не зрозуміло наскільки детальним він має бути і який результат можна вважати наближеним до людського опису зображення.

Першою успішною моделлю, яка дала початок активному розвитку цієї сфери, була модель Show and Tell [11]. Ідея цієї моделі полягає у поєднанні двох нейронних мереж — згорткової і рекурентної. Оскільки згорткові нейронні мережі активно застосовувались для задачі класифікації зображень і показували хороші результати, їх у даній моделі використовують на першому етапі для знаходження всіх зображених об’єктів. У моделі Show and Tell як згорткова нейронна мережа для класифікації була обрана Inception v3, описана нами у попередньому розділі. Отримавши ці об’єкти за допомогою Inception v3 і передавши їх в рекурентну нейронну мережу можна отримати текстовий опис зображення. Схему моделі Show and Tell показано на рис. 2.9.

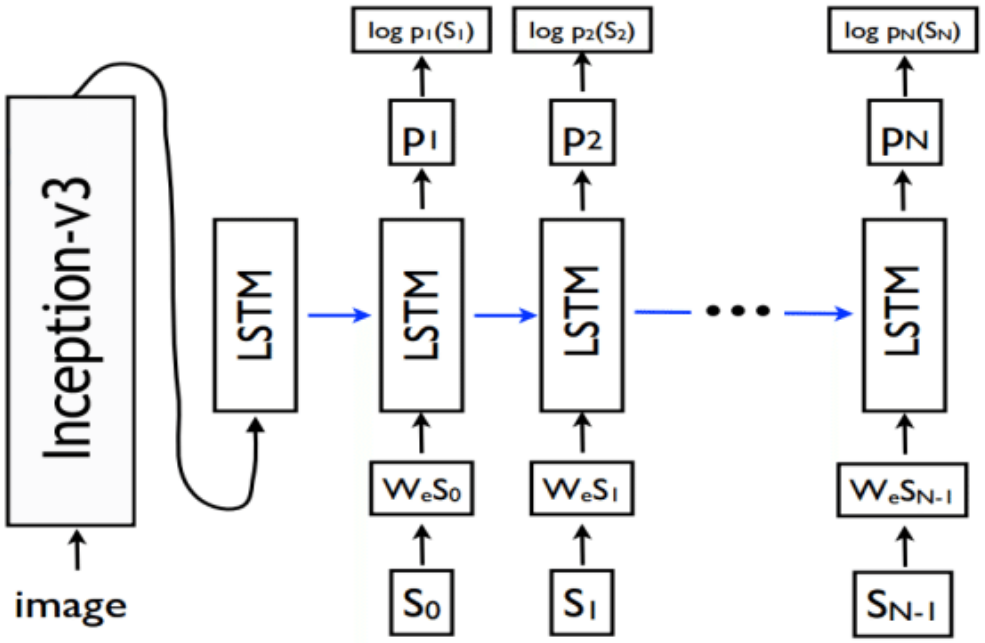


Рис. 2.9. Схема моделі Show and Tell [12]

Show and Tell мала значний успіх , і через це багато дослідників почали створювати її модифіковані варіанти. В результаті якість роботи цієї моделі було покращено і були знайдені нові методи. Останні розробки показують значно кращі результати: на сьогодні найкращою моделлю згідно з даними з сайту Paperwithcode [13] модель Oscar на наборі даних COCO captions за показником BLEU-4 (bilingual evaluation understudy, алгоритм оцінки якості тексту, який був перекладений машинно з однієї природної мови на іншу), що дорівнює 41.7, та так само займає перше місце у цій області за іншими показниками.

Текстовий опис загалом показав непогані результати та зараз активно використовується для вирішення проблеми опису зображень. Проте, новіші роботи у цій області показують неефективність опису зображення одним реченням.

З цього можна зробити висновок, що хоча текстовий опис і не є найкращим представленням семантичної моделі зображення, саме він допоміг розвинутись деяким іншим напрямкам і стати основою для вирішення цієї

проблеми. Тому у наступному підрозділі ми розглянемо який підхід врешті решт використовується наразі для створення текстового опису зображень та чому він вважається більш ефективним.

2.4 Знаходженння зв’язків на зображенні

При розгляді проблеми опису зображень ми згадували про те, що для розуміння семантики зображення окрім об’єктів на зображені важливі відносини (або зв'язки) між ними. Задача знаходження зв’язків на зображенні (visual relationship detection) якраз займається вирішенням цього питання. Метою цієї задачі є саме передбачення зв’язків між об’єктами на зображенні. Саме виявлення візуальних стосунків може допомогти подолати розрив між комп'ютерним зором і природною мовою для розуміння сцени зображень.

Під зв’язками у даному випадку мається на увазі як прийменники, так і дієслова. Ці зв'язки можуть описувати:

- як об’єкти розташовані відносно один одного (“під”, “поряд”, “наступний після”);
- яка дія відбувається між об’єктами (“йде”, “стоїть на”, “тримає”, “спостерігає за”);
- приналежність одних об’єктів до інших (“має”, “всі мають”);
- та багато інших.

Може скластися враження, що передбачення зв’язків не має сенсу, адже якщо результатом роботи є тільки зв’язки, то їх не можна інтерпретувати без об’єктів до яких ці зв’язки відносяться. Але для визначення об’єктів моделі з даної області користуються розробками в галузі пошуку об’єктів (object detection). Якщо поєднувати рішення цих двох напрямків можна отримати повноцінну інформацію про семантику зображення та, відповідно, побудувати його семантичну модель.

Загалом ми будемо розглядати цю задачу як задачу пошук триплетів (суб’єкт - зв’язок - об’єкт) незалежно від того, чи буде використовувався детектор об’єктів. У такому випадку задача позбавляється проблеми відсутності структурованості та неформальності, яка була при створенні текстового опису з результатом у вигляді одного речення. У нашому випадку результату буде чітко визначений у формі триплетів і його буде легко інтерпретувати, через що його можна буде широко використовувати надалі для вирішення інших задач.

Надалі буде наведений короткий аналіз останніх розробок у області знаходження зв’язків на зображені. Скориставшись його результатами ми визначимо, які розробки будуть найбільш актуальним для нас і що з них ми зможемо взяти за основу у свої роботі.

2.4.1 Recognition Using Visual Phrases

Першою спробою вирішити проблему знаходження зв’язків між об’єктами на зображені була робота під назвою Recognition Using Visual Phrases [14]. Ідея цієї роботи полягає в тому щоб розглядати задачу знаходження зв’язків як задачу класифікації. У цій роботі кожен триплет розглядається як окремий клас, тобто класами могли бути триплети “людина поруч із велосипедом”, “людина, що лежить на дивані” або “стрибки коня та вершника”. Ці класи були названі авторами роботи “візуальними фразами”.

Ідея використання візуальних фраз полягала у тому, що за допомогою них можна було надати більше інформації, ніж просто про один об’єкт, і менше ніж за всю сцену (вся множина об’єктів і їх зв’язків на зображенні) [14]. Оскільки передбачення всієї сцени є занадто складною задачею, а передбачення тільки одного класу не вирішує дану проблему, авторами роботи було прийнято рішення робити передбачення проміжного етапу між ними — візуальних фраз.

Для визначення конкретних візуальних фраз тренувалися окремі детектори, і так само робилося для кожного об’єкта. Потім під час тестування,

всі ці детектори застосовувались до вхідного зображення для отримання візуальних фраз і об’єктів . Ці результати об'єднували і передавали далі, де об’єкти без зв’язків видалялися, а ті що мали неповні доповнювались за допомогою попередньо визначених візуальних фраз. В результаті роботи моделі отримувались кілька візуальних фраз, оскільки кожен детектор міг визначити багато об’єктів. Схему роботи моделі наведено на рис. 2.10.

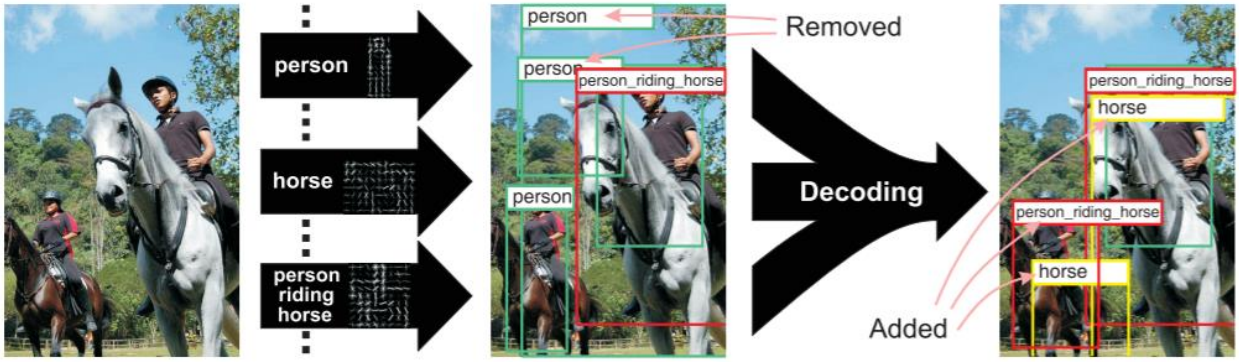


Figure 2. We use visual phrase and object models to make independent predictions. We then combine the predictions by a decoding algorithm that takes all detection responses and decides on the final outcome. Note that a) Visual phrase recognition works better than recognizing the participating objects. For example, the horse detector does not produce reliable predictions about horses in this picture while the “person riding horse” detector finds one instance; b) Our decoding then successfully adds two examples of horses and removes two wrong predictions of people by looking at other detections in the vicinity.

Рис. 2.10. Схема моделі Recognition Using Visual Phrases [14]

Недоліком моделі розробленій у цій роботі є те, що через те що для кожної візуальної фрази потрібно натреновувати новий детектор — через це час тренування моделі буде експоненційно збільшувати зі збільшенням кількості об’єктів. Через це такий підхід є сенс застосовувати тільки для не різноманітних наборів даних через його складність O(K*N^2). У цій роботі розглядалось тільки 17 візуальних фраз, навчальна вибірка для яких була створена ручним методом через пошуку зображень по цим фразам у пошуковій системі та подальшому фільтруванні результатів. Таким чином для кожної візуальної фрази було знайдено 50 зображень для навчання.

Попри це все, ідея візуальних фраз була першою спробою у цій області вирішити проблему знаходження зв’язків на зображенні і її результати доволі часто використовуються у порівнянні з новими моделями.

2.4.2 Visual Relationship Detection With Language Priors

Розроблена у лабараторії Стенфорда, робота Visual Relationship Detection With Language Priors [15] стала наступною віхою у цій області. Багато ідей з цієї роботи надалі почали широко використовуватися та покращуватися в наступних роботах.

Автори роботи явно порівнюють власне рішення з попередньо описаною роботою і пропонують свою модель як вирішення та покращення існуючих проблем. Першим покращенням відносно попередньої роботи є намір розглядати кожну візуальну фразу не як окремий клас. Натомість автори пропонують тренувати окремі класифікатори для об'єктів і зв'язків. Таким відбувається суттєве покращення, оскільки замість складності O(K*N^2) ми маємо складність O(K+N).

Також замість тренування детекторів для кожного об'єкта і кожного зв'язку, тренуються два класифікатори на основі згорткових нейронних мереж. Ця частина була названа авторами роботи візуальним модулем.

Також у цій роботі є ще один мовний модуль, який використовує модель word2vec [16]. Він займається покращенням результатів роботи моделі, а також для передбаченням об'єктів і зв'язків, яких явно може не бути в навчальній вибірці напочатку.

У фінальному результаті об'єднуються передбачення цих двох модулів. Також обидва модулі отримують пари об'єктів з детектора R-CNN [17]. Схема запропонованої моделі зображена на рис. 2.11.

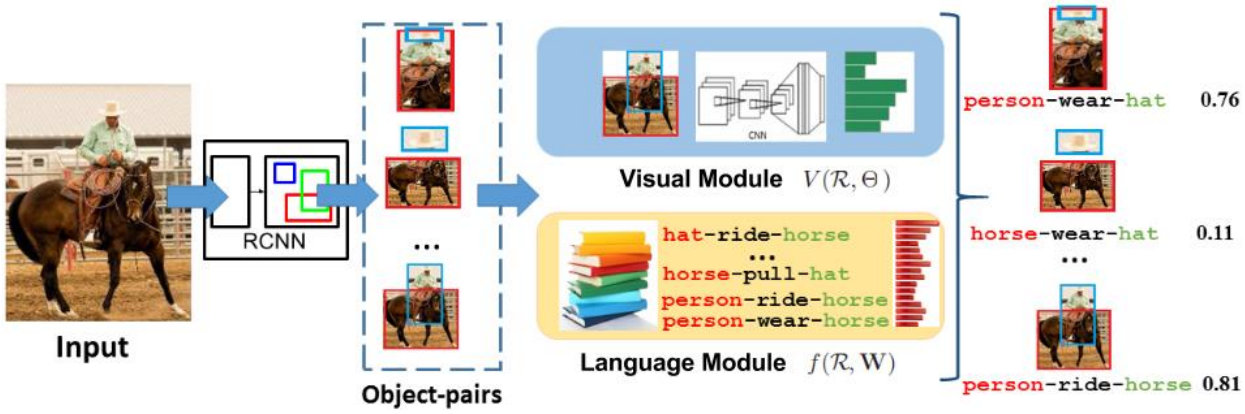


Рис. 2.11. Схема моделі Visual Relationship Detection With Language Priors [15]

Для оцінки якості роботи моделі використовується нова метрика “повнота з К” (Recall@K). За цією метрикою також оцінюється якість роботи моделі Visual Phrases та кілька модифікацій моделі. Останнє робиться для того щоб визначити наскільки кожен з компонентів даної системи впливає на фінальний результат.

Підсумовуючи, можна сказати що у цій роботі було зроблено кілька важливих внесків для вирішення задачі знаходження зв'язків на зображення, а саме було використано:

- згорткові нейронні мережі
- існуючий детектор для отримання об’єктів
- мовний модуль

Також у цій роботі було введено метрику “повнота з К” для оцінки якості роботи моделі, що теж є важливим внеском у вирішення задачі.

2.4.3 Detecting Visual Relations with Deep Relational Networks

Тепер, коли ми оглянули роботи які представляють собою основу у галузі визначення зв’язків на зображенні є сенс оглянути більш нещодавні. Однієб з них є робота під назвою Detecting Visual Relations with Deep Relational Networks [18]. Ця робота використовує ідеї своїх попередників та пропонує власні покращення.

В основі цієї роботи лежить ідея Deep Relational Network — моделі, яка використовує статистичні залежності між об’єктами і їх зв’язками. Автори провели дослідження, яке показало що є сильна залежність між об’єктами і зв’язками якими вони пов’язані та іншими зображеними об’єктами. Після дослідження було зрозуміло, що у більшості випадків між певною парою об’єктів може існувати тільки певна невелика кількість класів зв’язків, у той час як інші зв’язки зовсім не будуть притаманні цій парі. Це є доволі логічним висновком, оскільки навряд чи існують зображення де, наприклад, вікно буде розташовано на небі, а не на стіні якоїсь будівлі.

Через це було зроблено припущення, що при достатньо великому наборі вхідних даних для навчання, в результаті під час тестування моделі на нових зображення об’єкти будуть зустрічатись зі схожими зв’язками і на нових зображеннях приблизно з такою ж самою частотою. Дійсно, якщо якийсь зв’язок часто виникає між двома об’єктами, можна зробити висновок що ці два об’єкти часто перебувають у цьому зв’язку не лише у навчальній вибірці, але й на нових зображеннях.

Отже ці статистичні залежності між об’єктами і зв’язками можна зібрати з навчальної вибірки і надалі використовувати для передбачення певного класу зв’язку між двома об’єктами. Але все одно цей підхід виступає як допоміжний засіб у запропонованій моделі. Через це ці статистичні залежності тільки поширюються по моделі, а самі значення параметрів навчаються за допомогою позиційних рис і візуальних (зовнішніх) рис знайдених на зображенні.

Позиційні риси використовуються для передбачення зв'язків між об’єктами, які отримуються в залежност від того як об’єкти взаєморозташовані. Ці риси закокодуються за допомогою бінарних масок і потім обробляються модулем, представленному у вигляді згорткової нейронної мережі. Візуальні риси також обробляються окремим модулем, що так само представлений згортковою нейронною мережею.

В результаті позиційні і візуаліні риси об’єднуються через повнозв’язний шар. У кінці статистичні риси покращують передбачення і в результаті модель видає наймовірніші триплети суб’єкт - предикат - об’єкт для зображень. Схема моделі зображена на рис. 2.12.

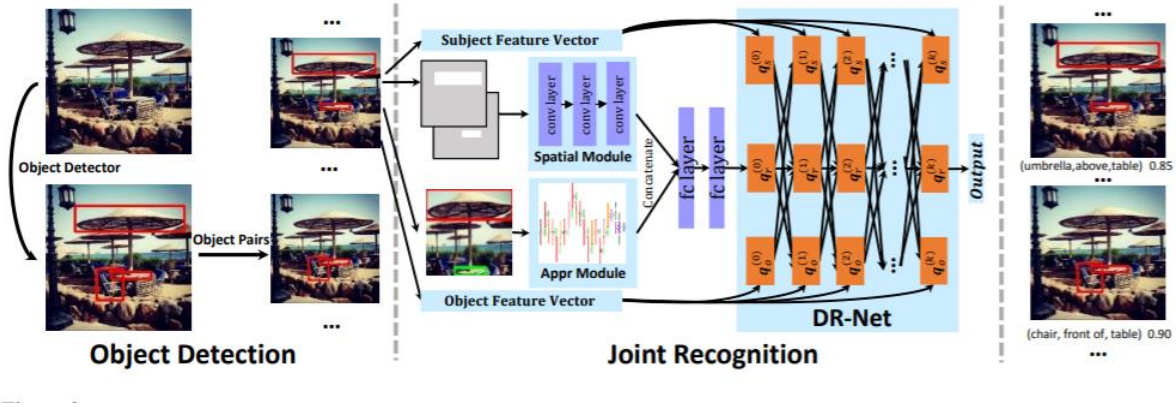


Рис. 2.12. Схема моделі Detecting Visual Relations with Deep Relational Networks [18]

2.4.4. Visual Relationship Detection with Internal and External Linguistic Knowledge Distillation

Одна з останніх розроблених моделей для вирішення задачі знаходження триплетів на зображенні є Visual Relationship Detection with Internal and External Linguistic Knowledge Distillation. І наразі, відповідно до даних з сайту Paperwithcode, вона досі займає перше місце у рейтингу моделей які найкраще вирішують задачу знаходження зв’язків за двома показниками Recall@100 та Recall@50, що становлять 31.89 та 22.68 відповідно [19].

Для вирішення нашої задачі автори роботи вирішили використати сильні кореляційні зв'язки між предикатом та парою суб’єкта та об’єкта (як семантично, так і просторово), щоб передбачати предикати, обумовлені даними суб'єктами та об'єктами.

Хоча моделювання трьох сутностей спільно точніше відображає їхні стосунки, це ускладнило навчання моделі, оскільки семантичний простір візуальних відносин є величезним, а навчальні дані обмежені. Щоб подолати

цю проблему, автори використали дані лінгвістичної статистики для того щоб регулювати навчання візуальних моделей.

Для цього було використані зв’язки та їх частота з навчальної вибірки (внутрішні знання) та загальнодоступний текст, з таких ресурсів як Вікіпедія, (зовнішні знання). Обидва джерела були використані для визначення, наскільки ймовірно що певний предикат буде з’єднувати певну пару об’єкта та суб’єкта. Ці дані надалі передавалися у глибоку нейронну мережу для досягнення кращого узагальнення.

Завдяки такому підходу модель досягла значних покращень і, наприклад, покращила метрику Recall з попередніх 8,45% до 19,17% на наборі тестування VRD zero-shot.

2.4.5 Context-Dependent Diffusion Network for Visual Relationship Detection

У цій роботі автори пропонують контекстно-залежну дифузійну мережу (CDDN) для вирішення задачі виявлення візуальних відносин. Для фіксації взаємодій різних екземплярів об’єктів побудовані два типи графіків — семантичний графік слова та графік візуальної сцени для розуміння взаємозалежностей по всьому зображенню.

Семантичний графік будується за допомогою мовних пріоритетів для моделювання семантичних кореляцій між об'єктами, тоді як графік візуальної сцени визначає зв'язки об'єктів сцени з метою використання навколишньої інформації про сцену зображення. Для графічно структурованих даних автори роботи розробляють дифузійну мережу для адаптивного агрегування інформації з контекстів, яка може ефективно засвоїти приховані подання візуальних відносин та добре забезпечити виявлення візуальних відносин з огляду на її ізоморфну незмінність до графіків.

На рис. 2.13. наведено архітектуру моделі CDDN.

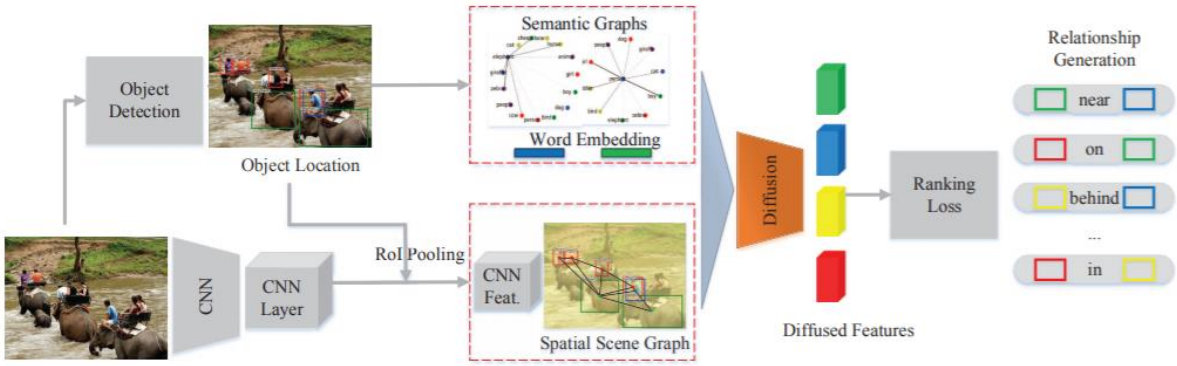


Рис. 2.13. Архітектура моделі CDDN [20]

Експерименти на двох широко використовуваних наборах даних (Visual Genome та Visual Relationship Dataset) демонструють, що запропонований авторами спосіб є більш ефективним і забезпечує state-of-the-art показники. У цій роботі було достягнуто показників 87.57 та 93.76 для R@50 та R@100 відповідно по визначенню предикатів, та 21.46 і 26.14 для R@50 R@100 відповідно по знаходженню візуальних стосунків на зображені [20].

Беручи до уваги опис підходу до вирішення задачі трьох останніх моделей, можна зробити висновок що для покращення результатів у цій області слід звернути увагу саме на частину задачі що створює текстовий опис.

Розділ 3. Розробка моделі

Розглянувши сучасні підходи до вирішення поставленої задачі, ми перейдемо до розробки власної моделі. Ми будемо використовувати деякі ідеї, розглянуті у попередніх розділах.

Наша модель буде отримувати на вхід зображення, а на виході повертати множину триплетів суб'єкт - предикат - об’єкт. Також ми будемо використовувати одну з існуючих навчальних колекцій, для навчання нашої моделі. Окрім цього, ми будемо спиратися на модель, наведену на цьому сайті [21].

3.1 Навчальна вибірка

Переглядаючи попередні роботи, що вирішують цю задачу, можна зрозуміти що найчастіше для знаходження візуальних зв’язків використовують два набори даних — Visual Genome (далі — VG) [22] та Visual Relationship Detection (далі — VRD) [15] (обидва розроблені Стенфордом). Обидві колекції містять в собі дані, що підходять для нашої задачі пошуку об’єктів та взаємозв’язків між ними.

Спочатку, разом з роботою [15] був створений набір даних VRD, для тренування моделі розробленої у рамках цієї роботи. Потім, як покращена та наступна версія, Стенфорд розробив VG, який вже містив набагато більше зображень, зв’язків та об’єктів. Якщо VRD містив 5000 зображень? 70 об’єктів та 70 предикатів, то у VG наразі містить 108 тис. зображень, 1.1 мільйон зв’язків та 108 тис. об’єктів.

Багато дослідників використовують модифікований варіант VG, оскільки його формат даних дещо надлишковий та містить багато шуму. Через це було вирішено обрати VRD, оскільки цей набір хоча й менший за розміром, але через це з ним простіше працювати і у нього немає таких недоліків, як у VG.

Для цієї роботи були обрані дані, відформатовані у pickle файли та відразу розбиті на навчальну, валідаційну та тестову вибірку [23].

Дані у VRD подаються у форматі, наведеному на рис 3.1.

```
{'2113966890_c65030a7e7_o.jpg': [{'object': {'bbox': [336, 489, 324, 458],
    'category': 5},
   'predicate': 9,
   'subject': {'bbox': [94, 175, 306, 590], 'category': 22}},
  {'object': {'bbox': [336, 489, 324, 458], 'category': 5},
   'predicate': 42,
   'subject': {'bbox': [94, 175, 306, 590], 'category': 22}},
  {'object': {'bbox': [94, 175, 306, 590], 'category': 22},
   'predicate': 15,
   'subject': {'bbox': [265, 397, 539, 680], 'category': 7}},
  {'object': {'bbox': [273, 394, 425, 556], 'category': 7},
   'predicate': 28,
   'subject': {'bbox': [265, 397, 539, 680], 'category': 7}},
  {'object': {'bbox': [270, 396, 302, 420], 'category': 7},
   'predicate': 28,
   'subject': {'bbox': [273, 394, 425, 556], 'category': 7}},
  {'object': {'bbox': [94, 175, 306, 590], 'category': 22},
   'predicate': 15,
   'subject': {'bbox': [273, 394, 425, 556], 'category': 7}},
  {'object': {'bbox': [272, 401, 159, 293], 'category': 7},
   'predicate': 28,
   'subject': {'bbox': [270, 396, 302, 420], 'category': 7}},
  {'object': {'bbox': [271, 407, 47, 156], 'category': 7},
   'predicate': 28,
   'subject': {'bbox': [272, 401, 159, 293], 'category': 7}},
  {'object': {'bbox': [272, 401, 159, 293], 'category': 7},
   'predicate': 28,
   'subject': {'bbox': [271, 407, 47, 156], 'category': 7}},
  {'object': {'bbox': [94, 175, 306, 590], 'category': 22},
   'predicate': 15,
   'subject': {'bbox': [336, 489, 324, 458], 'category': 5}}]}
```

Рис. 3.1. Формат даних колекції VRD

Як бачимо, для кожного зображення присутня назва його файлу, яка виступає також в ролі унікального ідентифікатора. Для перегляду зображень потрібно окремо завантажувати архів з ними. Також для кожного зображення присутній перелік зв’язків, у якому є суб’єкти (subject), предикати (predicate) та об’єкти (object). Для об’єктів та суб’єктів надані координати обмежувальних прямокутників (bbox), формат представлення яких є перелік координат ([ymin, ymax, xmin, xmax]). Також для всіх трьох надані ідентифікатори категорій, до яких вони відносяться.

Списки категорій об’єктів, суб’єктів та предикатів надається у окремих файлах. Приклад роботи з ними, а також приклади самих категорій можна побачити на рис. 3.2.

```
[44]  1  with open(file_location_vrd + 'json_dataset/objects.json') as f:
      2      vrd_objects = json.load(f)
      3
      4  print(len(vrd_objects))
      5  print(vrd_objects[0:5])

     100
     ['person', 'sky', 'building', 'truck', 'bus']

[45]  1  with open(file_location_vrd + 'json_dataset/predicates.json') as f:
      2      vrd_predicates = json.load(f)
      3
      4  print(len(vrd_predicates))
      5  print(vrd_predicates[0:5])

     70
     ['on', 'wear', 'has', 'next to', 'sleep next to']
```

Рис. 3.2. Приклад категорій об’єктів та предикатів з колекції VRD

Для наочності, ми візуалізуємо цю іноформацію на прикладі одного зображення з цієї колекції. Ми відобразимо зображенн разом з всіма його об’єкти та суб’єктами з їх класами. І також окремо, текстом, виведео зв’язки з цього зображення. Код для цієї операції наведено в лістингу 3.1, а саме зображення яке виводиться після виконання коду показано на рис. 3.3.

```python
image_id = at_img[741]

img = Image.open(file_location_vrd + 'sg_train_images/' + image_id).convert('RGBA')
triplets = [str(vrd_objects[an['subject']['category']]) + ' - ' +
            str(vrd_predicates[an['predicate']]) + ' - ' +
            str(vrd_objects[an['object']['category']]) for an in annotations_train[image_id]]
print('subject - predicate - object')
print(*triplets, sep = "\n")

boxes = Image.new('RGBA', img.size, (255,255,255,0))
boxes_draw = ImageDraw.Draw(boxes)

objects = [an['object']['bbox'] for an in annotations_train[image_id]]
objects_labels = [vrd_objects[an['object']['category']] for an in annotations_train[image_id]]
subjects = [an['subject']['bbox'] for an in annotations_train[image_id]]
subjects_labels = [vrd_objects[an['subject']['category']] for an in annotations_train[image_id]]

for i, box in enumerate(objects):
    boxes_draw.rectangle([box[2], box[0], box[3], box[1]], width=3, outline='red')
    x, y = box[2], box[0]
    label = objects_labels[i]
    w, h = boxes_draw.getfont().getsize(label)
    boxes_draw.rectangle((x, y, x+w, y+h), fill='black')
    boxes_draw.text((x, y), label)

for i, box in enumerate(subjects):
    boxes_draw.rectangle([box[2], box[0], box[3], box[1]], width=2, outline='blue')
    x, y = box[2], box[0]
    label = subjects_labels[i]
    w, h = boxes_draw.getfont().getsize(label)
    boxes_draw.rectangle((x, y, x+w, y+h), fill='black')
    boxes_draw.text((x, y), label)

Image.alpha_composite(img, boxes)
```

Лістинг 3.1. Код для візуалізації прикладу з навчальної вибірки даних

Рис. 3.3. Приклад візуалізації зображення з колекції VRD

Можна побачити, що такий формат даних є доволі зручним в роботі і підходить для нашої задачі.

3.2 Проектування архітектура моделі

Після того як ми вирішили, який у нас буде формат вхідних і вихідних даних та маючи навчальну вибірку, ми можемо приступити до проектуванням архітектури самої моделі. Звичайно, для нашої задачі основою нашої архітектури буде становити нейронна мережа. Оскільки ми будемо працювати з зображеннями, ми будемо використовувати згорткові неройнні мережі (далі

— ЗНМ). Оскільки на прикладі попередньо розглянутих робіт, можна зробити висновок що саме вони здебільшого використовуються для вирішення нашої задачі. До того ж, ЗНМ були розроблені якраз для оптимальних вирішень задач обробки зображень.

Оскільки, як ми побачили при аналізі існуючих рішень, є вже багато гарних моделей для підзадач, які нам буде потрібно вирішити, є сенс використати ці напрацювання. Для цього ми скористаємося підходом Transfer Learning.

3.2.1 Transfer Learning

Transfer Learning (TL) — це підхід в машинному навчанні, мета якого полягає в адаптуванні існуючих успішних моделей до нових задач. Наприклад, за цим підходом можна перенавчити існуючу модель на новому наборі даних, або вирішити схожу задачу за допомогою модифікації наявної моделі [24].

Моделі, що вирішують задачі комп'ютерного зору, під час тренування формують свою певну інтерпретацію зображення — певні візуальні або структурні риси зображеного. Для того, щоб модель могла показувати хороші результати, ці риси повинні бути досить універсальними, або інакше це призведуть до поганих результатів на невідомій новій вибірці під час тестування. З чого можна зробити висновок, що такі вивчені риси можна застосовувати до інших завдань, навіть якщо вони спочатку не передбачались при розробці даної моделі. Оскільки більшість завдань комп'ютерного зору спрямовані на краще розуміння зображення, TL добре працює для завдань цього типу.

Можна назвати ще одна перевагу на користь використання TL для нашої задачі. Проектування ЗНМ з нуля є досить складним завданням, оскільки нейронні мережі містять у собі багато гіперпараметрів для кожного прихованого шару. Більш того, кількість шарів мережі та їх архітектура так само можна назвати гіперпараметрами. Через стає майже неможливим створити свою, нову та кращу архітектуру ЗНМ. Тому зазвичай всі

використовують як основу успішні та попередньо натреновані моделі, які надали гарін результати для якогось базового завдання (наприклад, класифікації зображень) [24].

Тому для нашої роботи ми візьмемо дві попередньо натреновані моделі — Inception v3 (детальніший опис та переваги наведено у підрозділі 2.2.2.) та GloVe [25]. GloVe надає декілька файлів попередньо навчені векторів слів, і для нашої задачі ми оберемо найменший варіант — glove6B. Цей варіант містить слова з Wikipedia 2014 та Gigaword 5, та складається з 6 млрд. токенів, чого цілком достатньо для нашої задачі [25].

3.3 Розробка власної моделі

Inception v3 присутня у багатьох популярних бібліотеках та фреймворках машинного навчання, тому з нею просто працювати. Для нашої роботи ми оберемо бібліотеку Keras, і, відповідно, візьмемо цю модель звідти, а саме з бібліотеки keras.applications.

Для попереднього опрацювання навчальної вибірки та її завантаження у проект використовується бібліотека Pandas [26]. Код завантаження нашої вибірки наведено у лістингу 3.2.

```python
def vrd_to_pandas(relationships_set, objects, predicates):
    """Create Pandas DataFrame from JSON of relationships."""
    relationships = []

    for img in relationships_set:
        img_relationships = relationships_set[img]
        for relationship in img_relationships:
            relationships.append(flatten_vrd_relationship(img, relationship, objects, predicates))
    return pd.DataFrame.from_dict(relationships)


def load_vrd_data():
    relationships_train = json.load(open(file_location + "json_dataset/annotations_train.json"))
    relationships_test = json.load(open(file_location + "json_dataset/annotations_test.json"))
    objects = json.load(open(file_location + "json_dataset/objects.json"))
    predicates = json.load(open(file_location + "json_dataset/predicates.json"))

    np.random.seed(123)
    val_idx = list(np.random.choice(len(relationships_train), 1000, replace=False))
    relationships_val = {
        key: value
        for i, (key, value) in enumerate(relationships_train.items())
        if i in val_idx
    }
    relationships_train = {
        key: value
        for i, (key, value) in enumerate(relationships_train.items())
        if i not in val_idx
    }
    train_df = vrd_to_pandas(relationships_train, objects, predicates)
    valid_df = vrd_to_pandas(relationships_val, objects, predicates)
    test_df = vrd_to_pandas(relationships_test, objects, predicates)
    return train_df, valid_df, test_df
```

Лістинг 3.2. Завантаження навчальної вибірки у проект

Для завантаження векторів GloVe було використано код, наведений у лістингу 3.3.

```
[20]  # Load Glove vectors
      glove_dir = file_location
      embeddings_index = {}  # empty dictionary
      f = open(os.path.join(glove_dir, 'glove.6B.200d.txt'), encoding="utf-8")
      for line in f:
          values = line.split()
          word = values[0]
          coefs = np.asarray(values[1:], dtype='float32')
          embeddings_index[word] = coefs
      f.close()
      print('Found %s word vectors.' % len(embeddings_index))

      Found 400000 word vectors.
```

Лістинг 3.3. Завантаження вектора слів GloVe

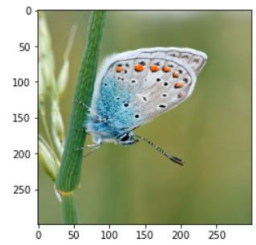
Також для завантаження навчальної вибірки вже саме у моделі Keras потрібно описувати взаємодію з даними вибірки через функцію data_generator.

Для нашої навчальної вибірки і моделі Inception v3, код цієї функції наведено в лістингу 3.4.

```python
# data generator, intended to be used in a call to model.fit_generator()
def data_generator(descriptions, subjects, objects, wordtoix, max_length, num_photos_per_batch):
    X1, X2, X3, y = list(), list(), list(), list()
    n = 0
    # loop for ever over images
    while 1:
        for k in range(len(descriptions)):
            n += 1
            # retrieve the photo feature
            sub = subjects[k]
            obj = objects[k]
            # for desc in descriptions:
            # encode the sequence
            seq = [wordtoix[word] for word in descriptions[k].split(' ') if word in wordtoix]
            # split one sequence into multiple X, y pairs
            for i in range(1, len(seq)):
                # split into input and output pair
                in_seq, out_seq = seq[:i], seq[i]
                # pad input sequence
                in_seq = pad_sequences([in_seq], maxlen=max_length)[0]
                # encode output sequence
                out_seq = to_categorical([out_seq], num_classes=vocab_size)[0]
                # store
                X1.append(sub)
                X2.append(obj)
                X3.append(in_seq)
                y.append(out_seq)
            # yield the batch data
            if n == num_photos_per_batch:
                yield [[array(X1), array(X2), array(X3)], array(y)]
                X1, X2, X3, y = list(), list(), list(), list()
                n = 0
```

Лістинг 3.4. Функція для завантаження навчальної вибірки

Приклад роботи моделі Inception v3 наведено на рис. 3.4.




```
Filename: butterfly_1.jpg
Displaying the top 5 Predictions for above image:
n02281787 lycaenid, lycaenid butterfly
n02277742 ringlet, ringlet butterfly
n02281406 sulphur butterfly, sulfur butterfly
n02280649 cabbage butterfly
n02276258 admiral
```

Рис. 3.4. Класифікація зображення за допомогою Inception v3 [27]

Оскільки задача цієї модель — класифікувати зображення, то з її допомогою можна отримати тільки одне передбачення, а не кілька, як у нашій

навчальній вибірці VRD. До того ж, Inception v3 використовує набір класів класи з колекції ImageNet, через що для того щоб отримати класи об’єктів, які присутні у нашій навчальній вибірці нам доведеться її перетренувати, що і буде застосування згаданого вище підходу TL.

В результаті перероблена архітектура нашої моделі була задана кодом, наведеним у лістингу 3.5., а її вигляд наведено на рис. 3.5.

```
[23]  1  inputs0 = Input(shape=(2048,))
      2  fe01 = Dropout(0.2)(inputs0)
      3  fe02 = Dense(256, activation='relu')(fe01)
      4
      5  inputs1 = Input(shape=(2048,))
      6  fe1 = Dropout(0.2)(inputs1)
      7  fe2 = Dense(256, activation='relu')(fe1)
      8
      9  inputs2 = Input(shape=(2048,))
     10  fe3 = Dropout(0.3)(inputs2)
     11  fe4 = Dense(256, activation='relu')(fe3)
     12
     13  inputs3 = Input(shape=(max_length,))
     14  se1 = Embedding(vocab_size, embedding_dim, mask_zero=True)(inputs3)
     15  se2 = Dropout(0.2)(se1)
     16  se3 = LSTM(256)(se2)
     17
     18  decoder1 = add([fe02, fe2, fe4, se3])
     19  decoder2 = Dense(256, activation='relu')(decoder1)
     20  outputs = Dense(vocab_size, activation='softmax')(decoder2)
     21
     22  model = Model(inputs=[inputs0, inputs1, inputs2, inputs3], outputs=outputs)
     23  model.summary()
```

Лістинг 3.5. Архітектура нашої моделі

Було додано чотири шари для обробки вхідних даних (inputs0, inputs1, inputs2, inputs3) та два для обробки результатів (decoder1, decoder2). Для всіх шарів була застосована функція активації ReLu, а для останнього — SoftMax. Також були додані шари Dropout для запобігання перетренування моделі. Ці шари випадково обнуляють вхідні дані з заданою частотою на кожному кроці під час навчання.

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_4 (InputLayer)            [(None, 9)]          0
__________________________________________________________________________________________________
input_1 (InputLayer)            [(None, 2048)]       0
__________________________________________________________________________________________________
input_2 (InputLayer)            [(None, 2048)]       0
__________________________________________________________________________________________________
input_3 (InputLayer)            [(None, 2048)]       0
__________________________________________________________________________________________________
embedding (Embedding)           (None, 9, 200)       32600       input_4[0][0]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 2048)         0           input_1[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 2048)         0           input_2[0][0]
__________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 2048)         0           input_3[0][0]
__________________________________________________________________________________________________
dropout_3 (Dropout)             (None, 9, 200)       0           embedding[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 256)          524544      dropout[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 256)          524544      dropout_1[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 256)          524544      dropout_2[0][0]
__________________________________________________________________________________________________
lstm (LSTM)                     (None, 256)          467968      dropout_3[0][0]
__________________________________________________________________________________________________
add (Add)                       (None, 256)          0           dense[0][0]
                                                                 dense_1[0][0]
                                                                 dense_2[0][0]
                                                                 lstm[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 256)          65792       add[0][0]
__________________________________________________________________________________________________
dense_4 (Dense)                 (None, 163)          41891       dense_3[0][0]
==================================================================================================
Total params: 2,181,883
Trainable params: 2,181,883
Non-trainable params: 0
__________________________________________________________________________________________________
```

Рис. 3.5. Параметри нашої моделі

Код для тренування моделі та результати перших епох наведено у лістингу 3.6.

```python
# Train model on training and optimize hyperparameter on validation dataset
epochs = 1000
number_pics_per_bath = 24
steps = len(train_descriptions)//number_pics_per_bath

for i in range(epochs):
    generator_train = data_generator(train_descriptions, train_subject_features,
                                     train_object_features, wordtoix, max_length, number_pics_per_bath)
    generator_val = data_generator(val_descriptions, val_subject_features, val_object_features,
                                   wordtoix, max_length, number_pics_per_bath)
    model.fit_generator(generator_train, epochs=1, steps_per_epoch=steps, verbose=1,
                        validation_data=generator_val, validation_steps=steps, callbacks=[es,tensorboard])
```

```
Epoch 1/1
951/951 [==============================] - 71s 75ms/step - loss: 3.4050 - accuracy: 0.3057 - val_loss: 3.2994 - val_accuracy: 0.3572
Epoch 1/1
951/951 [==============================] - 70s 73ms/step - loss: 2.8303 - accuracy: 0.3802 - val_loss: 2.8594 - val_accuracy: 0.3998
Epoch 1/1
951/951 [==============================] - 68s 72ms/step - loss: 2.5079 - accuracy: 0.4290 - val_loss: 2.5375 - val_accuracy: 0.4564
Epoch 1/1
951/951 [==============================] - 69s 72ms/step - loss: 2.2817 - accuracy: 0.4704 - val_loss: 2.2977 - val_accuracy: 0.4964
Epoch 1/1
951/951 [==============================] - 68s 72ms/step - loss: 2.1023 - accuracy: 0.5032 - val_loss: 2.0991 - val_accuracy: 0.5269
Epoch 1/1
951/951 [==============================] - 68s 71ms/step - loss: 1.9454 - accuracy: 0.5327 - val_loss: 1.8823 - val_accuracy: 0.5555
```

Лістинг 3.6. Тренування нашої моделі

3.4 Результати

Наша модель змогла досягти наступних результатів на наведених наборах даних:

- Під час тренування втрати досягнули показника у 0.6896, а точність становила 0.7892
- Під час валідації втрати досягнули показника у 0.7145, а точність становила 0.7132
- Під час тестування втрати досягнули показника у 1.1711, а точність становила 0.7365

На рис. 3.6. наведено графіки втрат та точності в залежності від кількості епох.

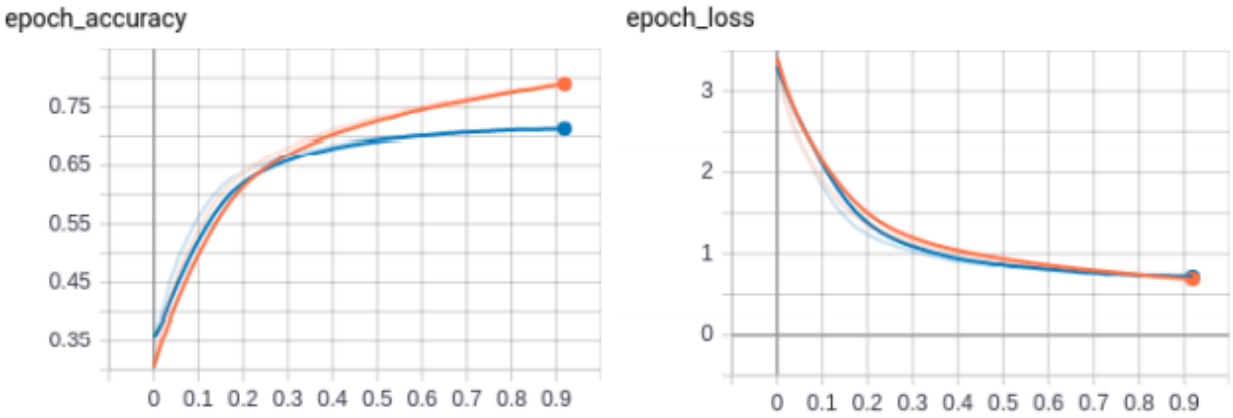


Рис 3.6. Графік точності та графік втрат для нашої моделі

Висновки

Задача знаходження зв’язків на зображенні є нелегкою та досить актуальною. На початку цієї роботи було доведено актуальність цієї задачі та розглянуто її проблематику. Надалі у цій роботі ми спробували зробити свій внесок у вирішення цієї задачі, попередньо проаналізувавши існуючі рішення та оцінивши їх результати. Було проаналізовано відомі моделі та моделі з найкращими показниками станом на квітень 2021 року для вирішення задач як класифікація зображення, текстовий опис зображення та виявлення зв’язків на зображені. Беручи до увагу висновки, зроблені на основі цього аналізу, була зроблена спроба спроектувати свою модель для виявлення зв’язків на зображеннях. Для цього був використаний підхід Transfer Learning, застосований до моделей InceptionV3 та GloVe. В результаті робота нашої моделі не досягло якихось значних показників та не змогла обійти зі своїми результатами state-of-the-art результати моделей, розроблених у рамках різних проаналізованих робіт, що вирішували ту саму задачу протягом останніх років.

Список використаних джерел

1. In the blink of an eye https://news.mit.edu/2014/in-the-blink-of-an-eye-0116
2. Data Never Sleeps 7.0 Infographic https://www.domo.com/learn/data-never-sleeps-7
3. Papers with Code - ImageNet Benchmark (Image Classification) https://paperswithcode.com/sota/image-classification-on-imagenet
4. VGG16 - Convolutional Network for Classification and Detection https://neurohive.io/en/popular-networks/vgg16/
5. Large Categories' Image Classifier - Inception v3 https://www.kaggle.com/imsparsh/large-categories-image-classifier-inception-v3
6. Understanding and visualizing ResNets | by Pablo Ruiz https://towardsdatascience.com/understanding-and-visualizing-resnets-442284831be8
7. Residual blocks — Building blocks of ResNet | by Sabyasachi Sahoo https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec
8. ResNet https://pytorch.org/hub/pytorch_vision_resnet/
9. EfficientNet: Improving Accuracy and Efficiency through AutoML and Model Scaling https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html
10. danieljl/keras-image-captioning: An implementation of image captioning in Keras https://github.com/danieljl/keras-image-captioning
11. Show and Tell: A Neural Image Caption Generator https://arxiv.org/pdf/1411.4555.pdf
12. Architecture of Show & Tell Model | Download Scientific Diagram https://www.researchgate.net/figure/Architecture-of-Show-Tell-Model_fig1_320957652

13. Image Captioning https://paperswithcode.com/task/image-captioning
14. Recognition Using Visual Phrases https://vision.cs.uiuc.edu/phrasal/recognition_using_visual_phrases.pdf
15. Visual Relationship Detection with Language Priors https://cs.stanford.edu/people/ranjaykrishna/vrd/
16. Efficient Estimation of Word Representations in Vector Space https://arxiv.org/pdf/1301.3781.pdf
17. Rich feature hierarchies for accurate object detection and semantic segmentation https://arxiv.org/pdf/1311.2524.pdf
18. Detecting Visual Relationships with Deep Relational Networks https://arxiv.org/pdf/1704.03114.pdf
19. Papers with Code - VRD Relationship Detection Benchmark (Visual Relationship Detection) https://paperswithcode.com/sota/visual-relationship-detection-on-vrd-1
20. Context-Dependent Diffusion Network for Visual Relationship Detection https://arxiv.org/pdf/1809.06213v1.pdf
21. Visual Relationship Detection https://github.com/AmanBudhraja/Visual-Relationship-Detection
22. VisualGenome https://visualgenome.org
23. visual_relation https://drive.google.com/drive/folders/10TxM6vc8xUrSxeiX8CnwD6YFNc1nYgGR?usp=sharing
24. CS231n Convolutional Neural Networks for Visual Recognition https://cs231n.github.io/transfer-learning/
25. GloVe: Global Vectors for Word Representation https://nlp.stanford.edu/projects/glove/
26. pandas - Python Data Analysis Library https://pandas.pydata.org
27. Classify Large Scale Images using pre-trained Inception v3 CNN model https://towardsdatascience.com/classify-any-object-using-pre-trained-cnn-model-77437d61e05f