Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра математики

Курсова робота

освітній ступінь – бакалавр

на тему: «**Поступове додавання класів для моделі виявлення об'єктів з архітектурою YOLO**»

Виконав: студент 4-го року навчання,
Спеціальності
122 «Комп'ютерні науки»

Боголій Владислав Валентинович

Керівник Швай Н. О.

«____» _____ 20____ р.

Київ – 2022

ГРАФІК ПІДГОТОВКИ КУРСОВОЇ РОБОТИ ДО ЗАХИСТУ

Графік узгоджено «___» _____ 20___ р.

№ з/п	Перелік робіт	Термін виконання	Дата ознайомлення наукового керівника	Підпис наукового керівника	Примітки
1.	Вибір теми, затвердження її на засіданні кафедри та закріплення наукового керівника Узгодження календарного графіка підготовки кваліфікаційної роботи. Ознайомлення студента з критеріями оцінювання кваліфікаційної роботи (п. 8.5).	жовтень			
2.	Вивчення джерел літератури, матеріалів архівів, періодичних видань, збір та узагальнення фактів, даних	жовтень – листопад			
3.	Складання плану каліф. роботи та узгодження з науковим керівником	листопад			
4.	Написання розділів роботи *або* Постановка експерименту, аналіз отриманих результатів наукового дослідження	листопад – березень			
5.	Проміжний контроль виконання роботи	лютий			
6.	Написання кваліфікаційної роботи в цілому, ознайомлення з її першим варіантом наукового керівника	січень – березень			
	Розділ 1 (постановка проблеми, теоретичні основи, огляд літературних джерел)				
	Розділ 2 (аналітично-дослідницька частина) (експериментальна частина для природничих і біологічних наук)				
	Розділ 3 (проектно-рекомендаційна частина) (аналіз результатів експерименту для природничих і біологічних наук)				
7.	Повне завершення написання кваліфікаційної роботи, оформлення її згідно з вимогами й подання на відгук науковому керівнику	квітень – початок травня			
8.	Подання кваліфікаційної роботи для перевірки письмових робіт студентів НаУКМА на відповідність вимогам академічної доброчесності,	середина травня			
9.	Подання на зовнішню рецензію				
10.	Підготовка до захисту кваліфікаційної роботи на засіданні кафедри: написання доповіді та виготовлення ілюстративного матеріалу	**до ____ травня**			
11.	Попередній захист кваліфікаційної роботи на засіданні кафедри	**до ____ травня**			
12.	Подання кваліфікаційної роботи на кафедру з усіма супроводжувальними документами	**до ____ травня**			
13.	Публічний захист кваліфікаційної роботи перед екзаменаційною комісією	згідно з розкладом роботи ЕК			

Науковий керівник _____ (ПІБ)

Виконавець курсової роботи _____ (ПІБ)

Зміст

Анотація

В даній роботі розглянуто методи додавання класів, для проблеми виявлення об’єктів, до існуючої моделі в умовах, коли дані для старих класів відсутні, або до них немає доступу. Наведені результати експериментів, які перевірили правильність висунутих гіпотез.

Вступ

В сучасному світі, зі збільшенням обчислювальної потужності процесорів, збільшенням кількості інформації на електронних носіях, став можливим швидкий розвиток в області глибоких нейронних мереж, зокрема класифікація зображень та виявлення об’єктів.

Актуальність теми:

Виявлення об’єктів застосовується для систем автоматичного керування машин, розпізнання об’єктів на супутникових знімках, розпізнавання ракових пухлин на знімках та в інших потребах. Навчання таких нейронних мереж є дорогим та тривалим у часі, тому додавання нових класів до існуючої нейронної мережі є набагато швидшим. До того ж, час навчання для додавання нових класів методами, які потребують для цього даних і для нових, і для старих класів, зростає лінійно до кількості цих даних, тому в цій роботі розглядається метод, що залежить лише від кількості нових даних.

Об’єкт дослідження:

Поступове додавання класів до нейронної мережі архітектури YOLO[1] в умовах відсутності навчальних даних для вже вивчених класів.

Розділ 1. Аналіз методів збільшення кількості класів

1.1. Навчання нової мережі

Найпростішим але і найбільш витратним з точки зору обчислювальних ресурсів способом збільшення класів є навчання нової мережі з потрібною кількістю вихідних класів. Для великих мереж такий спосіб є непрактичним, оскільки навчання займає багато часу.

1.2. Fine-tuning

Одним з поширених способів адаптації вже натренованих нейронних мереж для нових розпізнавання нових класів є fine-tuning, який полягає у відкиданні останніх шарів натренованої мережі, після чого до її залишку додаються кілька шарів з випадково ініціалізованими вагами. Далі модель тренується на нових класах з розповсюдженням градієнту лише на нові шари, тобто ваги вже натренованих шарів не зазнають змін. Для покращення результати кілька епох тренування проводять з розповсюдженням градієнту по всій мережі. Такий підхід дозволяє досягти високих результатів за меншу кількість ітерацій, порівняно з навчанням моделі з нуля [2]. Проте, цей метод, у випадку розширення натренованої мережі новими класами, вимагає навчання на вибірці, в якій присутня суттєва кількість представників кожного класу і є вкрай неефективним, якщо доступу до представників старих класів немає.

1.3. Дистиляція знань

Ця методика спочатку застосовувалася для передачі знань від великої натренованої моделі до меншої, з такою самою кількістю класів [3]. Суть ідеї полягає в припущенні, що, наприклад, для бінарної класифікації відповіді складної моделі у вигляді ймовірності несуть в собі більше інформації ніж просто істинне значення 1 або 0, тому у функцію втрат входять як істині значення, так і відповідь великої моделі. Також, в роботі зазначається, що для багато класової мережі слід використовувати функцію softmax з високою температурою, для більш м’якого розподілу ймовірностей та кращої передачі знань.

Для додавання нових класів з використанням даної техніки потрібно при тренуванні нової мережі з N+1 класами намагатися зберегти розподіл ймовірностей для старих N класів так, щоб він був наближеним для старої мережі з N класами. Дослідження показують, що такий метод є ефективним і дає змогу розширити кількість класів навіть за умови відсутності екземплярів для старих класів [4]. Цей метод покладений в основу даного дослідження.

1.4. Часткове тренування на старих класах

Для зменшення часу тренування при додаванні нових класів можна кластеризувати тренувальні дані та навчати модель лише на певній кількості екземплярів з кожного кластеру. У додаток до дистиляції знань такий підхід покращує точність моделі на старих класах [5]. Проте, у перспективі додавання багатьох нових класів, час тренування буде збільшуватися, також даний метод покладається на наявність даних для старих класів, що не завжди має місце.

Розділ 2. Тренування моделей

2.1. Середовище виконання

Для тренування моделей використовувалася платформа Kaggle, оскільки вона надає квоту в 30 годин на тиждень безкоштовного використання GPU для прискорення навчання.

2.2. Дані

Одними з найвідоміших наборів даних для задачі виявлення об’єктів є COCO та Pascal VOC. Оскільки COCO містить на порядок більше зображень, а також через обмеження на використання дискового простору платформи Kaggle, було вирішено провидити навчання на наборі даних Pascal VOC.

2.3. Реалізація архітектури YOLO

Основним завданням даного дослідження є пошук способу додавання нових класів, а не реалізація архітектури, тому для проведення експериментів було використано сторонню реалізацію архітектури YOLO, а саме YOLOv5[6]. Дана реалізація має конфігурації з різними розмірами, в усіх експериментах було використано конфігурацію YOLOv5s, яка має 7.2 мільйони параметрів.

2.4. Особливості навчання та функція втрат

При тренуванні моделі з n класами на даних з Pascal VOC, для пришвидшення навчання, її ваги були ініціалізовані вагами іншої мережі, що вже була натренована на наборі даних COCO. Цю мережу з n класами, що натренована на даних Pascal VOC, надалі будемо називати Mn.

Для розширення Mn була створена нова мережа, Mn+m, з розширеними останніми шарами, що обумовлено особливостями обраної реалізації та збільшенням кількості класів, які модель має розпізнавати. Всі шари Mn+m ініціалізуються вагами Mn, окрім розширень в останніх шарах. Під час навчання Mn+m використовуються зображення з мітками лише нових m класів, проте також використовується відповідь Mn для цих зображень, яка порівнюється з відповіддю Mn+m для початкових n класів.

Функція втрат має наступний вигляд:

$$L = \frac{1}{N}\sum_{i=0}^{N}\frac{1}{M_i}\sum_{j=0}^{M_i}\left(\frac{1}{A_j}\sum_{k=0}^{A_j}[bw \cdot lbox_k + ow \cdot lobj_k + cw \cdot lcls_k] + dw \cdot \frac{1}{B}\sum_{b=0}^{B} ldist_{ijb}\right)$$

- N – кількість зображень
- M_i – кількість міток (об’єктів) на зображенні i
- A_j – кількість anchor-box відповідальних за мітку j

- $lbox_k, lobj_k, lcls_k$ – результат функцій втрат для передбачених моделлю: локалізацій об'єкту, IoU (Intersection over Union), та класів відповідно
- bw, ow, cw, dw – ваги втрат, гіперпараметри
- B – загальна кількість anchor-box в моделі Mn+m, що відповідає кількості anchor-box моделі Mn
- $ldist_{ijb}$– результат дистиляційної функції втрат між моделлю Mn та Mn+m

$$lbox_i = 1 - iou_i$$

- iou_j – обрахований IoU справжньою локалізацією об'єкта, та передбаченою моделлю

$$lobj_i = BCEWithLogitsLoss(pobj_i, iou_i)$$

- $BCEWithLogitsLoss$ – [7]
- $pobj_i$ – IoU передбачений моделлю
- iou_i – обрахований IoU між справжньою локалізацією об'єкта, та передбаченою моделлю

$$lcls_i = \frac{1}{C}\sum_{j=0}^{C} BCEWithLogitsLoss(pcls_{ij}, cp \cdot tcls_{ij} + cn \cdot (1 - tcls_{ij}))$$

- C – кількість класів моделі Mn+m
- $BCEWithLogitsLoss$ – [7]

- $pcls_{ij}$ – передбачена моделлю ймовірність, що об’єкт належить до класу j
- cp, cn – згладжування істинної мітки для позитивних (тих, що належать даному класу) і негативних (тих, що не належать даному класу) екземплярів відповідно, гіперпараметри
- $tcls_{ij}$ – 1, якщо об’єкт насправді належить класу j, 0 в іншому випадку

$$iou_i = CIoU(xytrans(px_i), xytrans(py_i), whtrans(pw_i) \cdot aw_i, whtrans(ph_i) \cdot ah_i, tx_i, ty_i, tw_i, th_i)$$

- CIoU – покращена версія IoU[8]
- px_i, py_i, pw_i, ph_i – передбачення моделі для координат, ширини та висоти об’єкта
- tx_i, ty_i, tw_i, th_i – справжні координати, ширина та висота об’єкта
- aw_i, ah_i – відносні ширина та висота anchor-box

$$xytrans(p) = sigmoid(p) \cdot 2 - 0.5$$

$$whtrans(p) = (sigmoid(p) \cdot 2)^2$$

$$ldist_{ijk} = \frac{1}{N}\sum_{b=0}^{N} dbw \cdot dlbox_{ijkb} + dow \cdot dlobj_{ijkb} + dcw \cdot dlcls_{ijkb}$$

- N – кількість клітинок YOLO-сітки, які були відфільтровані певним чином в залежності від експерименту.

- dbw, dow, dcw – ваги втрат, гіперпараметри
- $dlbox_{ijkb}, dlobj_{ijkb}, dlcls_{ijkb}$ – результат функцій втрат для відфільтрованих передбачень моделлю Mn та Mn+m: локалізацій об’єкту, IoU (Intersection over Union), та класів відповідно

$$dlbox_{ijkb} = (pxold_{ijkb} - pxnew_{ijkb})^2 + (pyold_{ijkb} - pynew_{ijkb})^2 + (pwold_{ijkb} - pwnew_{ijkb})^2 + (phold_{ijkb} - phnew_{ijkb})^2$$

- $pxold_{ijkb}, pyold_{ijkb}, pwold_{ijkb}, phold_{ijkb}$ – передбачення моделі Mn для координат, ширини та висоти об’єкта
- $pxnew_{ijkb}, pynew_{ijkb}, pwnew_{ijkb}, phnew_{ijkb}$ – передбачення моделі Mn+m для координат, ширини та висоти об’єкта

$$dlobj_{ijkb} = ((pobjold_{ijkb} - meanoldo_{ijk}) - (pobjnew_{ijkb} - meannewo_{ijk}))^2$$

- $pobjold_{ijkb}, pobjnew_{ijkb}$ – IoU між передбаченою та справжньою локалізацією об’єкта, передбачені моделями Mn та Mn+m
- $meanoldo_{ijk}, meannewo_{ijk}$ – середнє значення передбачень моделями Mn та Mn+m IoU між справжньою локалізацією об’єкта та передбаченням моделей, для всіх відфільтрованих клітинок anchor-box k

$$dlcls_{ijkb} = \frac{1}{C}\sum_{d=0}^{C}\left((pcold_{ijkbd} - meanoldc_{ijk}\right) - (pcnew_{ijkbd} - meannewc_{ijk}))^2$$

- $pcold_{ijkbd}, pcnew_{ijkbd}$ – вірогідності, що об’єкт належить класу d, передбачені моделями Mn та Mn+m
- $meanoldc_{ijk}, meannewc_{ijk}$ – середнє значення передбачень моделями Mn та Mn+m ймовірностей для всіх класів, для всіх відфільтрованих клітинок anchor-box k

Розділ 3. Результати експериментів

Всі моделі в наведених нижче експериментах тренувалися 10 епох. Для оцінки моделей використовувалась метрика mAP@.5[9]. Тестування здійснювалося на зображеннях перших 10ти класів набору даних Pascal VOC. Результаті всіх експериментів наведені в таблиці 3.1. Літера b в назвах моделей від box, o – від objectness, c – від class, w – від weight.

Модель	Старі класи mAP@.5	Нові класи mAP@.5	Всі класи mAP@.5
M9	0.827	–	0.827
M10	0.83	–	0.83
M9+1_freeze	0.071	0.024	0.066
M9+1	0.071	0.52	0.116
M9+1_boc	0	0	0
M9+1_bc	0.47	0.25	0.448
M9+1_bc_w2	0.542	0.2	0.508
M9+1_bc_w4	0.286	0.0272	0.26
M9+1_c	0.311	0.447	0.324
M9+1_bc_o>0.5	0.12	0.535	0.162
M9+1_c_o>0.5	0.125	0.504	0.162
M9+1_bc_k5	0.171	0.444	0.198
M9+1_bc_k5_w10	0.438	0.264	0.42
M9+1_bc_k5_w50	0.283	0.244	0.279

Таблиця 3.1. Результати навчання

Спочатку було натреновано модель на перших дев’яти класах Pascal VOC. Ця модель має назву M9 в таблиці 3.1, приклад передбачення моделі можна побачити в додатку А, рисунок 2 (надалі буде позначатися А.n та буде означати додаток А, малюнок n). Також для порівняння були натреновані моделі M10 – навчання на 10ти класах, приклад передбачень на рисунку А.3, та M9+1 – донавчання моделі M9 на даних лише нового класу та без дистиляційної функції втрат. На рисунку А.4 можна побачити, що модель M9+1 добре справляється з локалізацією, але втратила здібність класифікувати старі класи. Модель M9+1_freeze містить заморожені шари моделі M9 і була натренована лише на даних, що мають мітки нового класу.

Наступною, була спроба навчити M9+1_boc, де застосовувалася дистиляційна функція втрат для локалізації, передбачення IoU та передбачення класів. Модель показала нездатність до вивчення нового класу та втрату знань про вже вивчені класи – рисунок А.5.

Після кількох експериментів, було зроблено припущення, що модель M9+1_boc має такі результати через вплив передбачення IoU M9 в клітинках, де насправді немає об’єкту, тому вага уподібнення між IoU старої моделі та нової була виставлена як 0. Після цього нова модель M9+1_bc вперше показала прийнятний результат. Проте на рисунку А.6 видно, що проблеми стосовно класифікації все ще залишаються.

Після відносно вдалого навчання M9+1_bc, була спроба збільшити вагу дистиляційної функції в 2 рази та натренувати M9+1_bc_w2. Це, як і очікувалося, покращило точність моделі на старих класах, але погіршило вивчення нового. На рисунку А.7 видно, що модель почала краще розпізнавати старі класи, однак часто плутає їх з новим через те, що після передбачення відфільтровуються локалізації з високим IoU (non-maximal suppression) лише для однакових класів. Також була спроба ще збільшити

вагу дистиляційної функції та натренувати M9+1_bc_w4 (рисунок А.8), але це не дало покращення точності.

Наступною була модель M9+1_c, в якій враховувалась лише дистиляція передбачень для класів (рисунок А.9). Це покращило середню точність на новому класі, але погіршило на старих.

Наступним було припущення, що найбільш важливими є результати старої моделі з високим IoU, тому в моделях M9+1_bc_o>0.5 та M9+1_c_o>0.5 дистиляція відбувалася лише для передбачень, в яких передбачений IoU старої моделі був більшим за 0.5.

Інша гіпотеза стосувалася того, що обмежена кількість передбачень для кожного з anchor-box має допомогти фокусуватися на найбільш підходящих об’єктах в кожному anchor-box. Тому в моделях M9+1_bc_k5, M9+1_bc_k5_w10 та M9+1_bc_k5_w50 було відібрано 5% клітинок YOLO-сітки, з найбільшим IoU, передбаченим моделлю M9, для цих клітинок застосовувалася дистиляційна функція втрат.

Висновки

В цій роботі було розглянуто функцію втрат та перевірені підходи щодо додавання нових класів до моделі з архітектурою YOLO. Результати експериментів показують, що деякі підходи, наприклад як для моделі M9+1_bc, показують набагато кращу точність ніж заморожування шарів та звичайне навчання на лише нових класах, хоча ця точність і менша ніж тренування точність моделі натренованої на всіх класах.

Проблема додавання нових класів в умовах відсутності даних для старих є непростою та потребує подальшого дослідження, адже її вирішення значно пришвидшить додавання класів у порівнянні з навчанням нової моделі з більшою кількістю класів.

Список використаних джерел

1. Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi – You Only Look Once: Unified, Real-Time Object Detection (2016): https://arxiv.org/pdf/1506.02640.pdf
2. Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik – Rich feature hierarchies for accurate object detection and semantic segmentation (2014): https://arxiv.org/pdf/1311.2524.pdf
3. Geoffrey Hinton, Oriol Vinyals, Jeff Dean – Distilling the Knowledge in a Neural Network (2015): https://arxiv.org/pdf/1503.02531.pdf
4. Konstantin Shmelkov, Cordelia Schmid, Karteek Alahari – Incremental Learning of Object Detectors without Catastrophic Forgetting (2017): https://arxiv.org/pdf/1708.06977.pdf
5. Dawei Li, Serafettin Tasci, Shalini Ghosh, Jingwen Zhu, Junting Zhang, Larry Heck – RILOD: Near Real-Time Incremental Learning for Object Detection at the Edge (2019): https://arxiv.org/pdf/1904.00781.pdf
6. YOLOv5 [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/ultralytics/yolov5
7. BCEWithLogitsLoss [Електронний ресурс] – Режим доступу до ресурсу:https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html
8. Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, Dongwei Ren – Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression (2019): https://arxiv.org/pdf/1911.08287.pdf
9. Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, Andrew Zisserman – The PASCAL Visual Object Classes (VOC) Challenge: https://homepages.inf.ed.ac.uk/ckiw/postscript/ijcv_voc09.pdf

Додатки

Додаток А

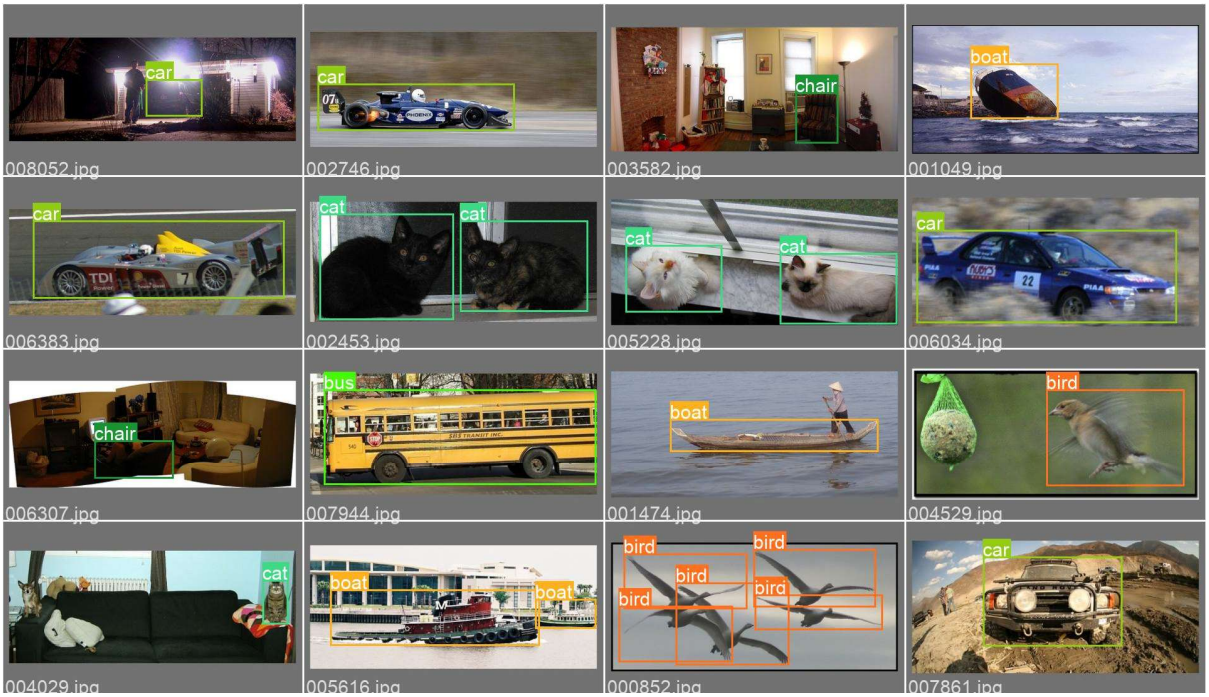

Рисунок 1. Справжні мітки

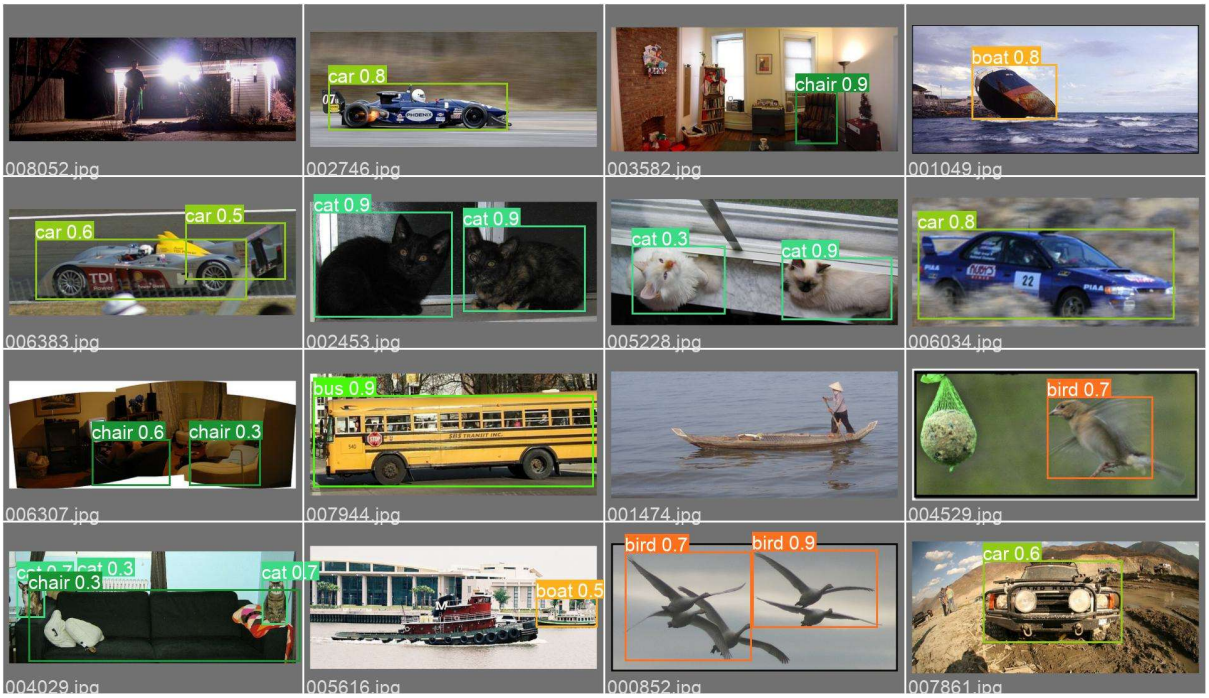

Рисунок 2. Передбачення моделі M9

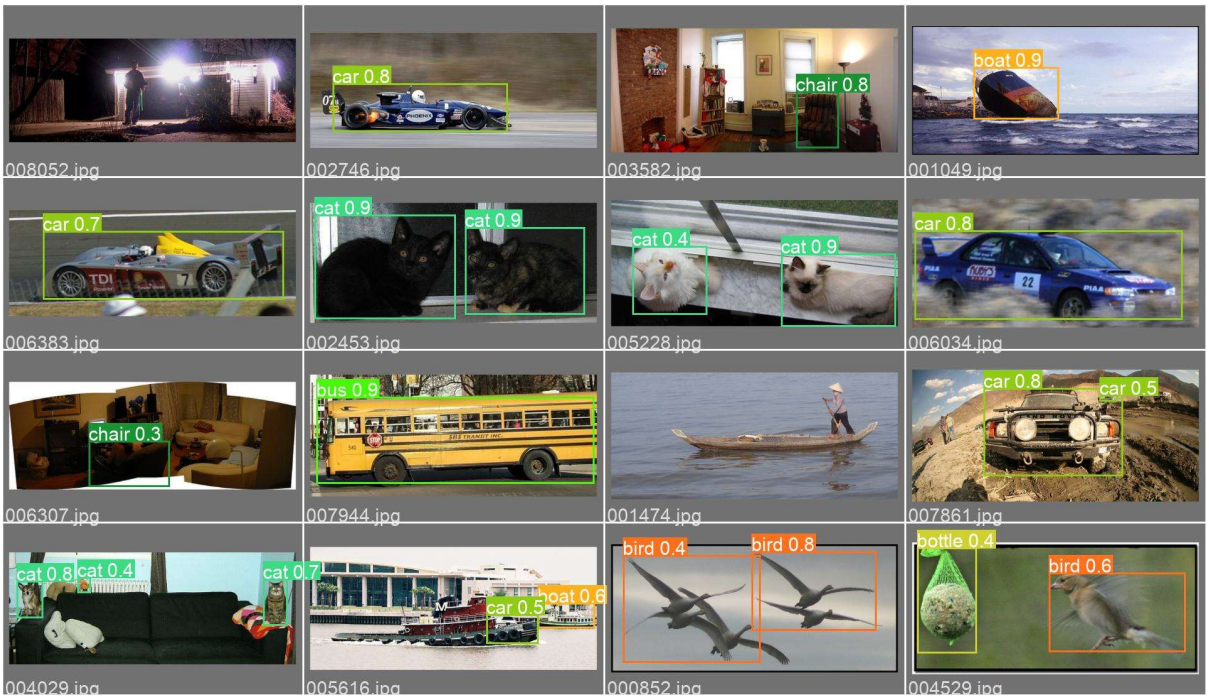

Рисунок 3. Передбачення моделі M10

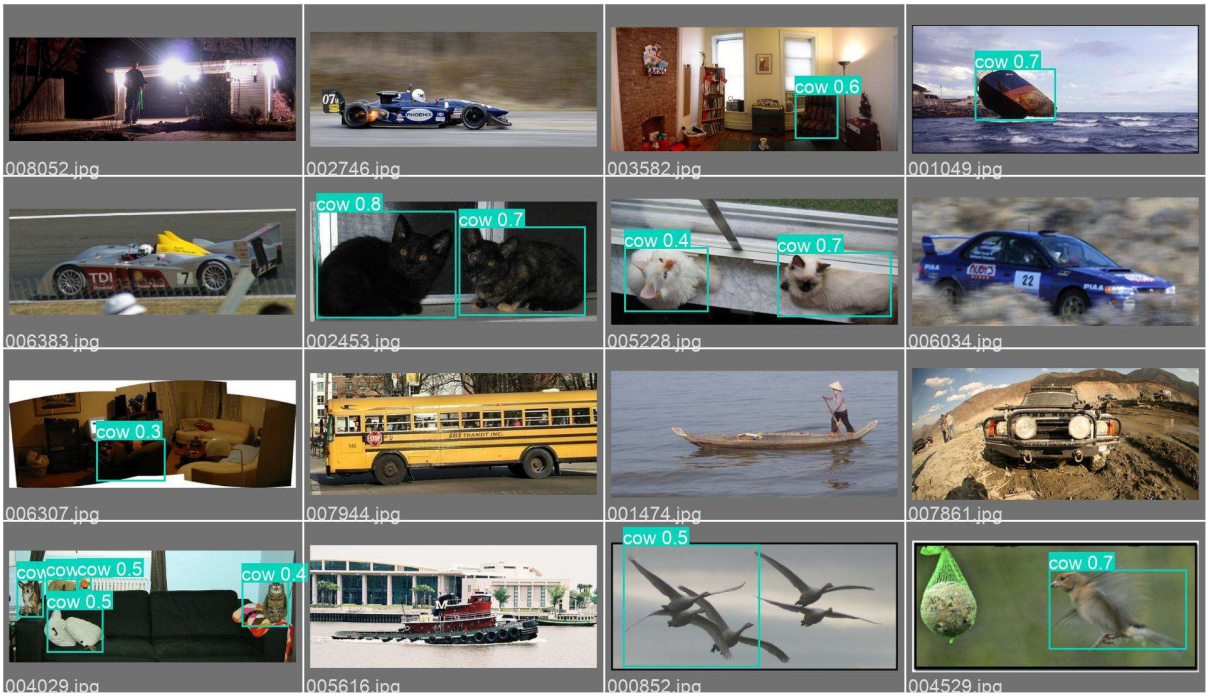

Рисунок 4. Передбачення моделі M9+1

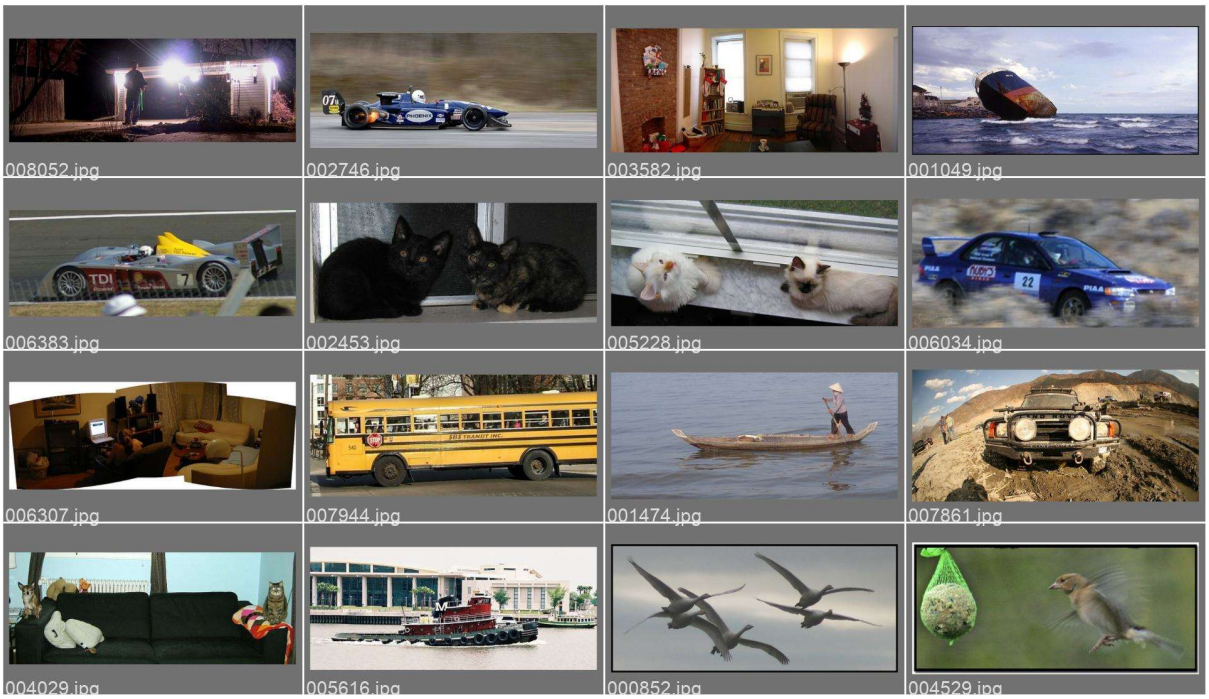

Рисунок 5. Передбачення моделі M9+1_boc

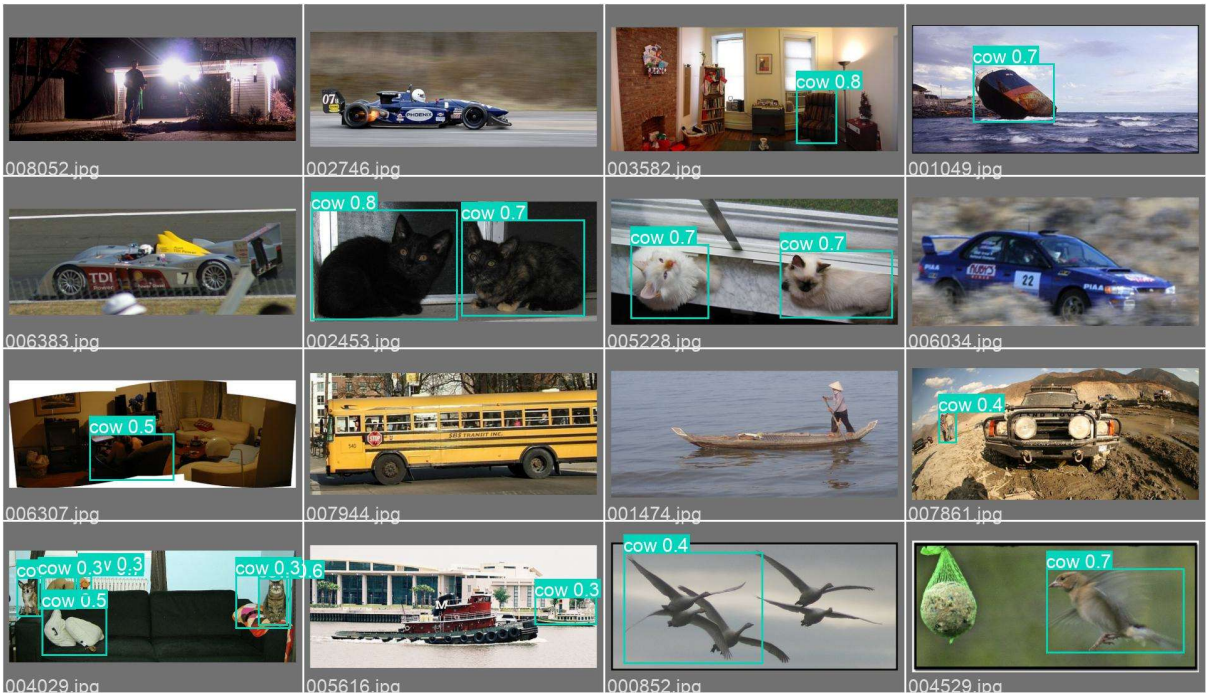

Рисунок 6. Передбачення моделі M9+1_bc

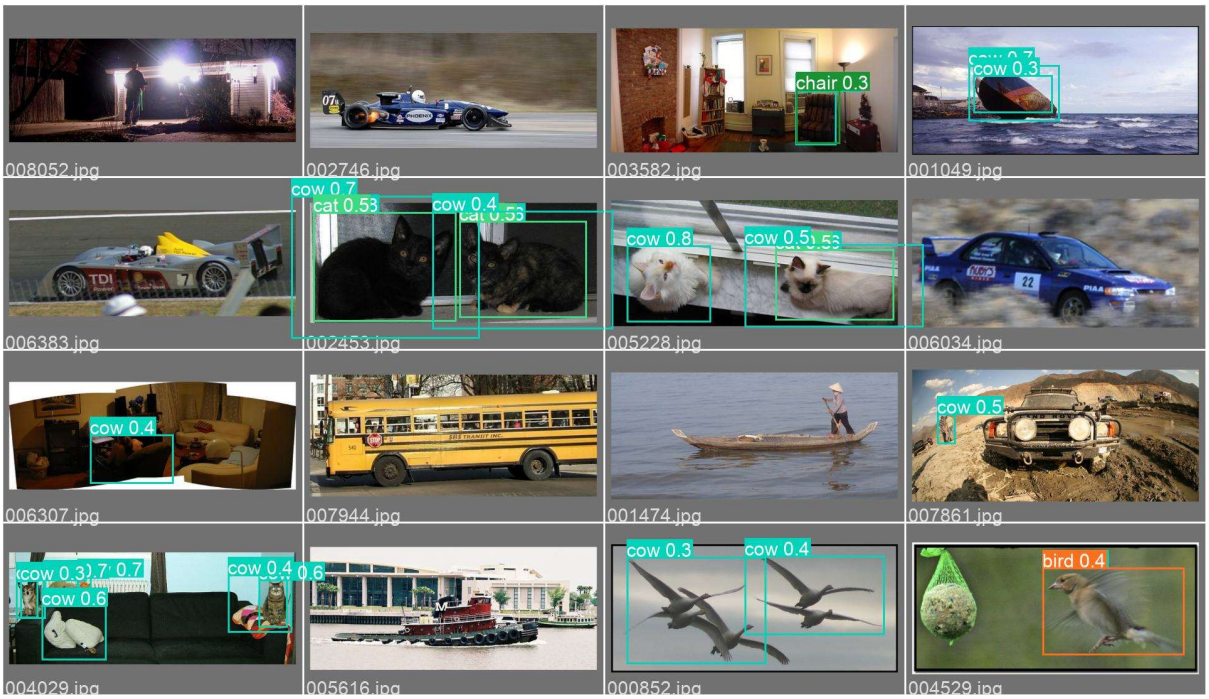

Рисунок 7. Передбачення моделі M9+1_bc_w2

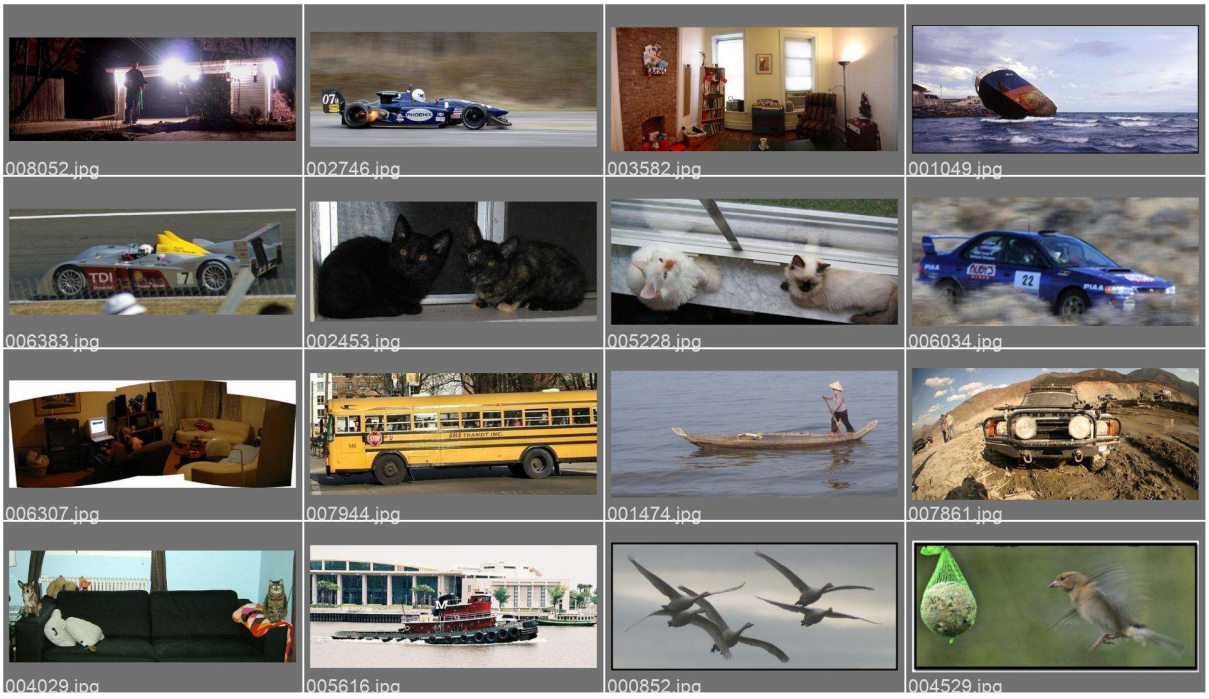

Рисунок 8. Передбачення моделі M9+1_bc_w4

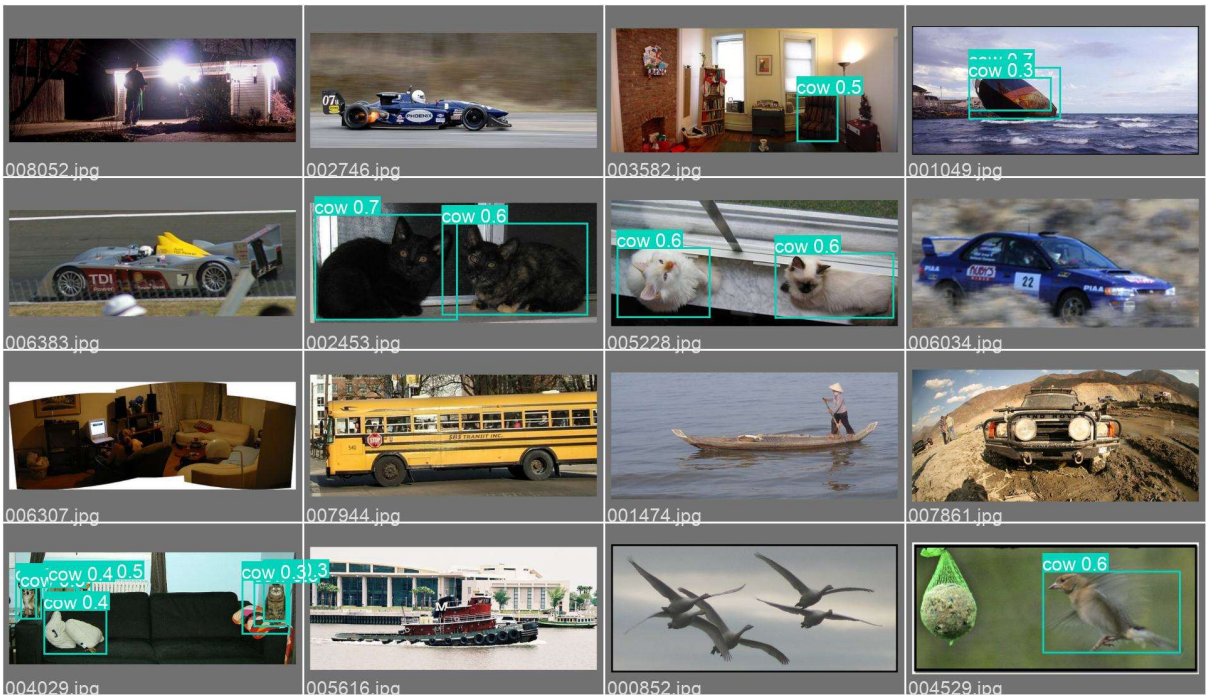

Рисунок 9. Передбачення моделі M9+1_c

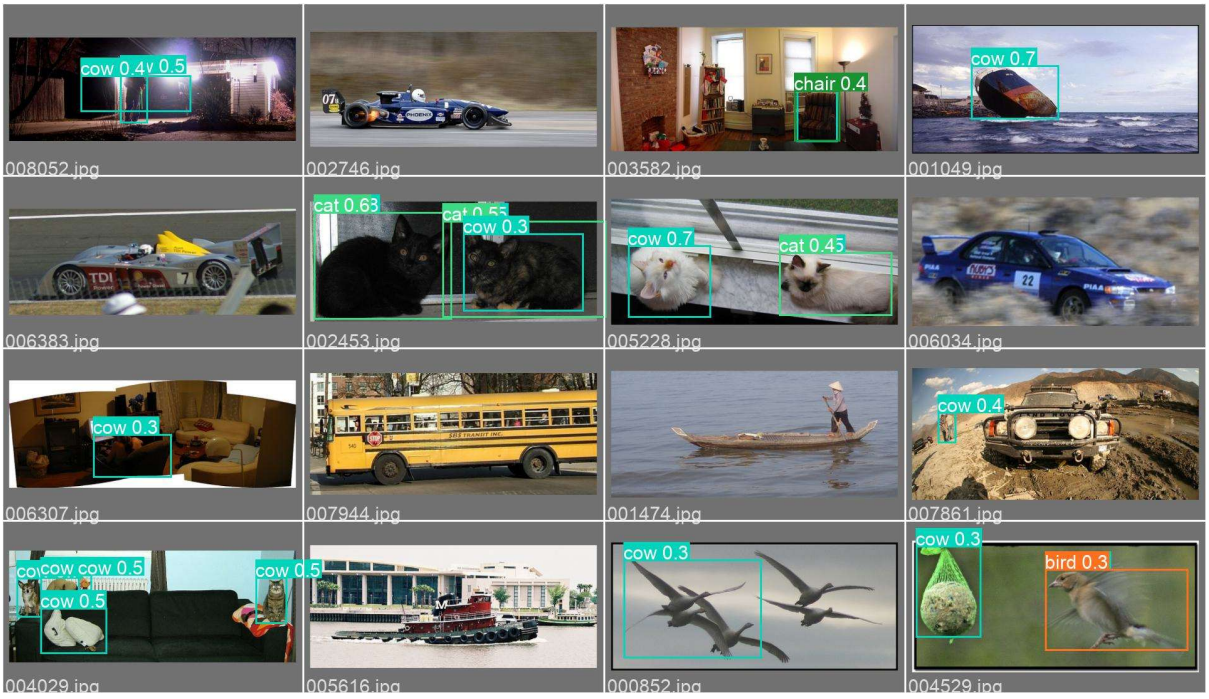

Рисунок 10. Передбачення моделі M9+1_bc_k5_w10