УДК 004.85

1. Модель класифікації пошкоджень на склі автомобіля за допомогою згорткових нейронних мереж
2. Славінський В.О.

Вступ

Один з пріоритетних напрямків страхування на сьогоднішній день займає автострахування. Більшість випадків, які припадають на автострахування, потрапляють під категорії визначення пошкоджень на лобовому склі. За правилами цих пошкоджень існує два основних способи впливу при прийнятті рішень, такі як заміна лобового скла повністю або ремонт за допомогою страхових партнерів.

Страхова сфера є досить складною і має велику кількість різноманітних нюансів. Також, страхові компанії досить закриті і в наш час ще часто користуються застарілим програмним забезпеченням або паперовим видом оформлення страхових випадків.

**Дане дослідження** **присвячене** порівняльному аналізу найбільш популярних архітектур згорткових мереж [1], створенню моделей згорткової нейронної мережі та вибору найкращої моделі за метриками якості для вирішення практичної задачі класифікації пошкоджень скла автомобіля. Модель, отримана в результаті, значно полегшить роботу страховим компаніям у визначенні пошкоджень. Розроблена система не буде залежати від людського фактору, що позитивно вплине як на страхові компанії, так і на посередників.

Одна з ключових особливостей побудованої моделі глибокого навчання та програмного продукту на її основі – це можливість автоматичного визначення категорії пошкодження. Тому додаток може працювати незалежно від людей, які є носіями знань, а це принесе велику гнучкість в майбутньому масштабуванні на велику кількість страхових компаній та на посередників.

**Постановка задачі**

Нехай Х – множина опису об’єктів, Y – скінченна множина номерів (імен) класів. Існує невідома цільова залежність-відображення , значення якої відомі тільки на об’єктах скінченної вибірки . [2] Необхідно побудувати алгоритм: , здатний класифікувати довільний об’єкт .

Дано: розміченні зображення з пошкодженнями на склі автомобільних засобів ( рис.1).

Необхідно розробити систему, яка отримує фотографію на вхід, після цього користувач міг би виділити зону, на якій є пошкодження, а система відповіла, до якої з категорій це пошкодження належить: Repair (ремонт) або Replacement (заміна).

Изображение выглядит как окно

Автоматически созданное описание

Рисунок 1 Вхідні зображення пошкоджень

Вимоги до системи: система повинна працювати на сервері, щоб користувачу не було потреби завантажувати будь-які додаткові модулі та бібліотеки для роботи системи. Окрім цього, важливо, щоб різні компоненти не залежали один від одного, щоб можна було розробляти їх паралельно. Також система повинна мати зрозумілий програмний інтерфейс, щоб у майбутньому її можна було легко інтегрувати в іншу систему, що вимагає розпізнання пошкоджень.

Результатом роботи буде виведена на екран назва категорії пошкодження.

Множина розмічених зображень з пошкодженнями на склі автомобільних засобів, що задана, розбивається на три підмножини: тренувальна, валідаційна та тестова вибірки [2, 3]. Тренувальна – вибірка, завдяки якій відбувається налаштування (оптимізація) ваг моделі. Валідаційна – вибірка для підбору гіперпараметрів та ознак, що стосуються навчання моделі, зокрема для прийняття рішень щодо перенавчання моделі. Тестова – вибірка для оцінки якості навчання моделі.

Задачі класифікації зображень вирішуються, як правило, за допомогою методів штучної згорткової нейронної мережі  у вигляді навчання з учителем.

Короткий опис шарів згорткової нейронної мережі (CNN)

Архітектура CNN аналогічна структурі зв’язку нейронів в мозку людини і натхненна організацією зорової кори. Окремі нейрони реагують на подразники лише в обмеженій області зорового поля, відомому як рецептивне поле. Колекція таких полів накладається на всю зорову зону [4].

Один із найпопулярніших напрямків використання цієї архітектури – класифікація зображень. Наприклад, Facebook використовує CNN для алгоритмів автоматичного позначення, Amazon - для генерації рекомендацій щодо продуктів, а Google - для пошуку серед фотографій користувачів. [5]

Операція згортки зазвичай позначають зірочкою ( [6]:

|  |  |
| --- | --- |
|  |  |

Якщо розглядати дану операцію з точки зору нейронних мереж, то є вхідним значенням, а – ядром. Результат операції іноді називають картою характеристик [7].

Архітектури CNN складаються з окремих шарів. У всіх випадках шари приймають як вхідний 3D-об'єм, перетворюють цей об'єм за допомогою диференціальних рівнянь і виводять 3D-об'єм. Деякі шари вимагають налаштування гіперпараметрів, а інші - ні.

**Вхідний шар** (**Input layer)** представляє вхідні значення пікселів зображення у вигляді 3D-матрицірозмірності Ш х В х Г, де глибина Г відповідає кількості кольорових каналів зображення.

**Згортковий шар (Convolution layer)** обчислює вихідні дані вузлів, підключених до локальних областей вхідної матриці. В цьому шарі обчислюються точкові добутки між набором вагових коефіцієнтів, який зазвичай називають фільтром, та значеннями з локальної області введення.

Результат згорткового шару подається на поелементну функцію активації, як правило ReLU. **Шар активації (Activation layer)** визначає, чи буде вхідний вузол "працювати" з урахуванням вхідних даних. Так, якщо вузол "спрацьовує", то це означає, що фільтри шару згортки виявили візуальну особливість. Функція ReLU визначається як max (0, x) з пороговим значенням 0. Для зменшення ширини та висоти вихідного зображення застосовується об’єднання пікселів та відповідний **шар пулінгу або субдискретизації (Pooling layer)**.

Останніми у мережі CNN є повнозв’язні шари (**Fully connected layer).** Імовірності класів обчислюються, наприклад, з використанням функції SoftMax і виводяться в 3D-масиві з розмірами: [1x1xK], де K – кількість класів.

Також у мережі CNN часто застосовується форма регуляризації, яка називається випаданням (**Dropout**): випадковим чином відбираємо нейрони, які ігноруються під час навчання (рис.2). Випадання тимчасово усуває внески в активацію нейронів, а також оновлення ваг покращує моделі та перешкоджає певним наборам ваг спеціалізуватися на конкретних особливостях, що може призвести до перенавчання.

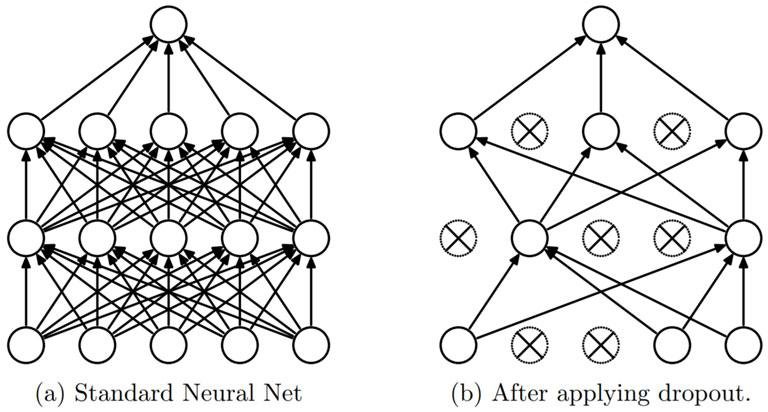


Рисунок 2 Приклад роботи шара випадання [8]

Особливості сучасних глибоких архітектур CNN

**Модель MobileNet** розроблена для використання в мобільних додатках, і це перша модель мобільного комп'ютерного зору TensorFlow. Особливість MobileNet у тому, що використовуються відокремлені поглиблені шари згортки. Це значно зменшує кількість параметрів у порівнянні з мережею зі звичайними шарами згортки з однаковою глибиною в сітках.

MobileNet – невелика модель з низькою затримкою та малою потужністю, параметризована для задоволення обмежень ресурсів у різних випадках використання для задач класифікації, виявлення та сегментації [9]. Цей клас CNN був розроблений для Google, і дає нам чудову відправну точку для навчання класифікаторів у нашій задачі, які мають бути доволі малими, але водночас швидкими.

**EfficientNet** – згорткова архітектура нейронної мережі та метод масштабування MB (рис.3), який на відміну від звичайної практики рівномірно масштабує всі розміри глибини/ширини/роздільної здатності завдяки набору фіксованих коефіцієнтів масштабування [10, 11]. Ідея складеного методу масштабування полягає у тому, що якщо вхідне зображення більше, то мережі потрібно більше шарів для збільшення сприйнятливого поля та більше каналів для захоплення більш дрібнозернистих візерунків на більшому зображенні.

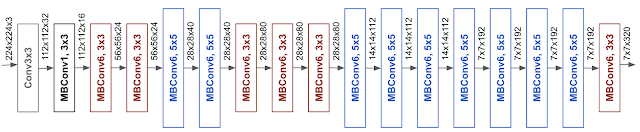


Рисунок 3 Архітектура EfficientNet [11]

**Мережа VGG-16** має 13 згорткових та 3 повнозв’язані шари, останній з них є шаром Softmax. Першою важливою відмінністю, яка стала фактичним стандартом для подальших глибоких архітектур – це використання великої кількості маленьких фільтрів, зокрема, розміром 3 × 3 та 1 × 1 з кроком рівним одиниці.

Шари Max-Pooling використовуються після більшості, але не після кожного згорткового шару, аналогічно прикладу AlexNet. Обґрунтуванням було те, що укладені згорткові шари з меншими фільтрами наближають ефект одного згорткового шару з фільтром більшого розміру. Наприклад, три складені згорткові шари з фільтрами 3 × 3 наближаються до одного згорткового шару з фільтром 7 × 7. Всі об'єднання у шарах Max-Pooling виконуються з розміром вікна 2 × 2 і тим самим кроком, що теж стало фактичним стандартом.

Інша важлива відмінність мережі VGG-16 – дуже велика кількість використовуваних фільтрів, і кількість фільтрів збільшується із глибиною моделі. На нижніх шарах мережі використовуються 64 фільтри і ця кількість збільшується до 128, 256 та 512 фільтрів на верхніх шарах моделі [12].

У **моделях Inception** ключовим нововведенням є модуль початку (іnception, звідси і назва архітектури). Це блок з трьох паралельних згорткових шарів з фільтрами різного розміру, такими як 1 × 1, 3 × 3, 5 × 5, та одного шару макс-пулінгу 3 × 3. Результати перерахованих чотирьох шарів потім об’єднуються.

Другим важливим дизайнерським рішенням у моделі Inception було підключення виходу в різних точках моделі. Це було досягнуто шляхом створення невеликих вихідних мереж з основної мережі, які були навчені робити прогнози. Метою було надати додатковий сигнал помилки для задачі класифікації в різних точках глибинної моделі для вирішення проблеми зникаючих градієнтів. Потім ці невеликі вихідні мережі були видалені після навчання.

**ResNet.** Зростання кількості шарів у архітектурі CNN може призводити до зниження її продуктивності. Причина полягає в тому, що глибші мережі набагато важче оптимізувати через проблему зникнення / вибуху градієнтів. Для збільшення кількості згорткових шарів, уникаючи проблеми вибуху / зникнення градієнтів, у 2015 році було винайдено ResNet (рис. 4, 5) [1]. В основі ResNet лежить ідея залишкового навчання. Найбільша кількість згорткових шарів моделі ResNet може бути більше 1000.

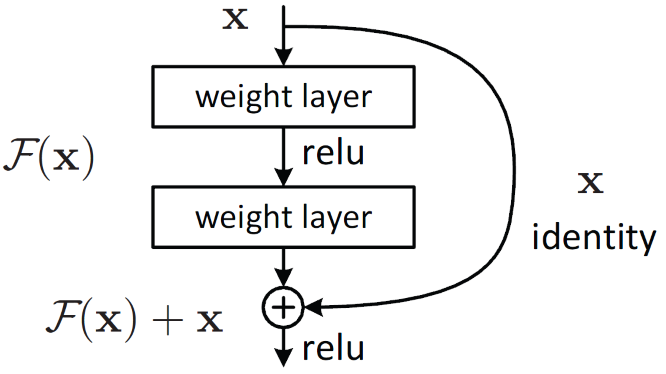


Рисунок 4 Залишкове навчання

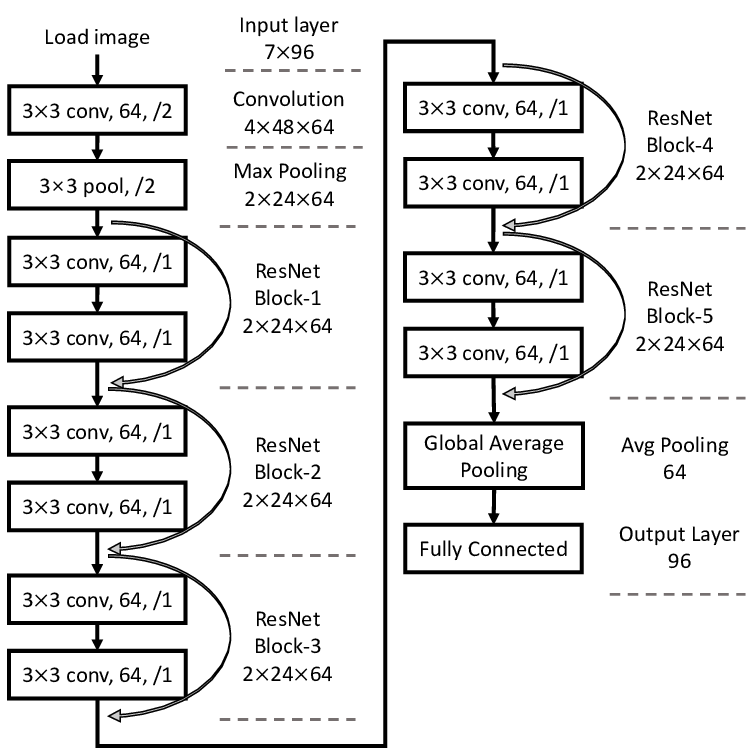


Рисунок 5 Архітектура ResNet

Інтуїтивно висновок кожного шару згортки є принаймні таким же гарним, як і вхідний сигнал. тобто F (x) + x ≥ x. Доведено, що ця архітектура добре вирішує проблему градієнта.

Розробка системи та веб додатку для класифікації зображень пошкоджень моделями нейронних мереж

Розроблена система складається з трьох глобальних компонентів: клієнта, сервера та моделей CNN. Розглянемо їх детальніше.

Першим та найважливішим компонентом системи є моделі згорткових нейроних мереж. Побудова та відбір моделей проводилися у системі Google Colaboratory, професійному середовищі програмування. Google Colaboratory – це безкоштовний хмарний сервіс на основі Jupyter Notebook. Основні переваги цього сервісу: не потребує довгої інсталяції та налаштування, дає доступ до графічних процесорів, які доволі коштовні, але набагато ефективніші, дозволяє легко інтегруватися з Google Drive, а це в свою чергу допомагає не бути прив’язаним до одного пристрою, має вбудовану історію змін.

**Веб-додаток** – це клієнт-серверний додаток, в якому клієнт взаємодіє з веб-сервером за допомогою браузера. Логіка веб-додатка розподілена між сервером і клієнтом, зберігання даних здійснюється, переважно, на сервері, а обмін інформацією відбувається по мережі. Одною з переваг такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, тому веб-додатки є міжплатформовими сервісами. Веб-додаток отримує запит від клієнта і виконує обчислення, після цього формує веб сторінку і відправляє її клієнтові мережею з використанням протоколу HTTP. [13]

Параметри, які реалізуються на стороні сервера: зберігання, захист і доступ до даних, обробка та валідація клієнтських запитів, відправка відповідей клієнту.

Параметри, які реалізуються на стороні клієнта: сторінка з інтерактивним графічним інтерфейсом, формулювання запиту до сервера і його подальша відправка, отримання відповіді від сервера та подальша її обробка. [14]

Фронтенд (frontend) або **клієнтська частина програми** виконується в браузері користувача. Ця частина написана на мові програмування Javascript. Додаток може складатися тільки з клієнтської частини, а то й потрібно зберігати дані користувача довше однієї сесії. Це можуть бути, наприклад, фоторедактори або прості іграшки.

Single page application (SPA або односторінкове додаток) – більш цікавий варіант, коли використовуються і бекенд і фронтенд. За допомогою їх взаємодії можна створити додаток, який буде працювати без перезавантажень сторінки в браузері. Або в спрощеному варіанті, коли переходи між розділами викликають перезавантаження, але будь-які дії в розділі обходяться без них. [15]

У роботі використано сервер Flask – це мікро-фреймворк, написаний на Python. Він класифікується як мікро-фреймворк, оскільки для нього не потрібні певні інструменти чи бібліотеки. Він не має рівня абстракції бази даних, перевірки форми або будь-яких інших компонентів, де вже існуючі сторонні бібліотеки забезпечують загальні функції. Однак Flask підтримує розширення, які можуть додавати функції програми так, ніби вони реалізовані в самому Flask. Саме тому він є гарним рішенням для інтеграції з TensorFlow моделями.

Порівняльний аналіз навчених моделей

Розглянемо різні моделі та проаналізуємо результати їх роботи, а після зробимо порівняльну таблицю, щоб зрозуміти, яка модель показала себе краще за всіх. Для всіх моделей умови запуску були однакові:

* Дані аугментовані на випадковий поворот та віддзеркалення,
* Функція втрат – SparseCategoricalCrossentropy,
* Оптимізатор – Адам,
* Learning rate – 3Е-4,
* Розмір батча – 64,
* Кількість епох – 80.

Хоча кількість епох була задана як 80, була використана технологія EarlyStopping. Вона дозволяє стежити за обраною метрикою, і якщо ця метрика не покращується за декілька епох, EarlyStopping зупиняє навчання мережі. Цей підхід дозволяє уникнути проблеми перенавчання.

Результати навчання моделей наведено на рис. 6 – 10. Для кожної з архітектур приведено два графіка залежностей значень від кількості епох. Перший – це графік точності (accuracy), другий – графік втрат (loss). На кожному з рис. 6 – 10 наведено дві криві: помаранчева представляє графік тренувальних даних, синя – графік валідаційних даних.

|  |  |
| --- | --- |
| Рисунок 6 Залежності значень точності (accuracy) та функції втрат (loss) від кількості епох під час навчання Inception | Рисунок 7 Залежності значень точності (accuracy) та функції втрат (loss) від кількості епох під час навчання моделі EfficientNet |

|  |  |
| --- | --- |
| Рисунок 8 Графіки навчання моделі ResNet | Рисунок 9 Графіки навчання моделі MobileNet |

|  |  |
| --- | --- |
|  |  |
| Рисунок 10 Графіки навчання моделі NasNet | |

Таблиця 3.1 – Результати навчання Inception

|  |  |  |  |
| --- | --- | --- | --- |
|  | Train | Validation | Test |
| Accuracy | 0.8225 | 0.8277 | 0.8441 |
| Loss | 0.3822 | 0.3662 | 0.3416 |

Таблиця 3.2 – Результати навчання EfficientNet

|  |  |  |  |
| --- | --- | --- | --- |
|  | Train | Validation | Test |
| Accuracy | 0.8434 | 0.8226 | 0.8584 |
| Loss | 0.3503 | 0.8434 | 0.3357 |

Таблиця 3.3 – Результати навчання ResNet

|  |  |  |  |
| --- | --- | --- | --- |
|  | Train | Validation | Test |
| Accuracy | 0.8410 | 0.8451 | 0.8471 |
| Loss | 0.3516 | 0.3387 | 0.3306 |

Таблиця 3.4 – Результати навчання MobileNet

|  |  |  |  |
| --- | --- | --- | --- |
|  | Train | Validation | Test |
| Accuracy | 0.8120 | 0.3886 | 0.8482 |
| Loss | 0.4002 | 0.8267 | 0.3491 |

Таблиця 3.5 – Результати навчання NasNet

|  |  |  |  |
| --- | --- | --- | --- |
|  | Train | Validation | Test |
| Accuracy | 0.8205 | 0.8226 | 0.8461 |
| Loss | 0.3915 | 0.3750 | 0.3537 |

Таблиця 3.6 – Порівняння різних архітектур за значеннями accuracy і loss на тестовій множині зображень

|  |  |  |
| --- | --- | --- |
| Архітектура | Test accuracy | Test loss |
| Inception | 0.8441 | 0.3416 |
| EfficientNet | 0.8584 | 0.3357 |
| ResNet | 0.8471 | 0.3306 |
| MobileNet | 0.8482 | 0.3491 |
| NasNet | 0.8461 | 0.3537 |
| **Краща** | **EfficientNet** | **ResNet** |

**Приклад роботи веб додатку**

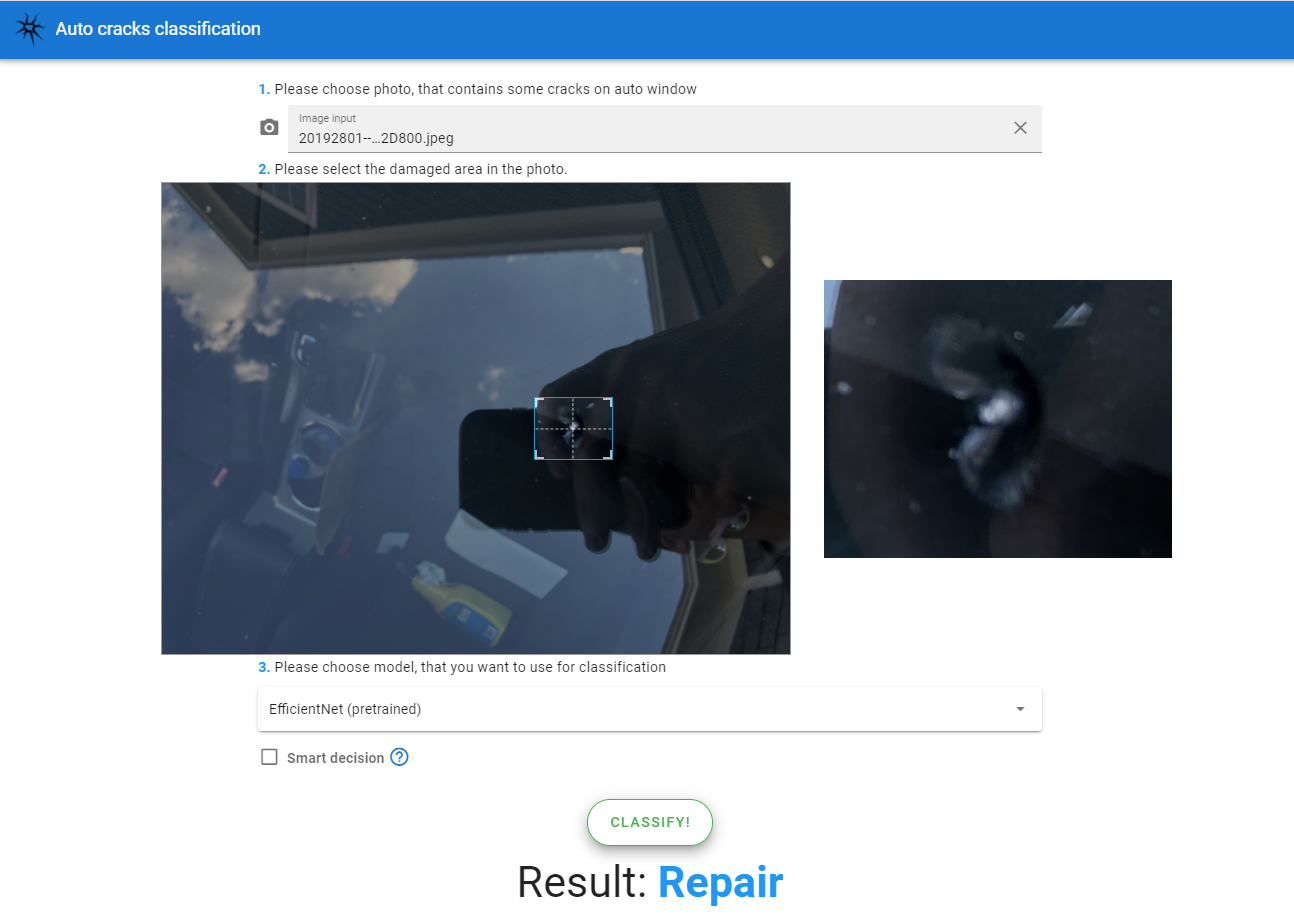


Рисунок 11 Приклад роботи веб додатку

Веб додаток складається з однієї сторінки. Для роботи ним потрібно по-перше завантажити фото, яна якому є пошкодження. Після того, як зображення буде завантажене, нам пропонується виділити на ньому область з пошкодженням.

Останнім пунктом є вибір моделі для класифікації. При кліку відкриється випадаючий список з усіма описаними вище моделями. Також можна обрати пункт «Розумний вибір», яка получить результати усіх моделей, після цього візьме середнє арифметичне виведе той результат, який більша кількість моделей обрала. Після усіх приготувань необхідно лише натиснути кнопку «Класифікувати!», яка відправить запит на сервер і отримає результат моделі.

Обговорення результатів

Оскільки була обрана попередньо навчена мережа, валідаційні графіки трохи кращі, ніж навчальні. Усі моделі показали себе більш менш однаково в навчанні та дали приблизно схожі результати. Графіки сходяться один до одного, що каже про гарне навчання мережі (рис. 6 – 10).

Inception, EfficientNet та NasNet показали стрімкий зріст за перші 3-4 епохи і покращилися на 5-8% (за метрикою точності). ResNet та MobileNet показали найбільш стабільне покращення (без швидких перепадів).

Усі мережі припинили своє навчання значно раніше 80-ї епохи, оскільки значення функції втрат не покращувалося за декілька епох. Таким чином ми уникнули проблеми перенавчання.

Висновки

У роботі побудувано моделі згорткових нейронних мереж для класифікації пошкоджень на склі автомобіля. Виконано порівняльний аналіз найбільш популярних архітектур згорткових мереж. Виконано попередній аналіз та обробку вхідних даних для подальшого використання у роботі. Обрано найкращі згорткові моделі за метриками якості для вирішення практичної задачі класифікації пошкоджень скла.

Розроблено систему та веб додаток для зручної класифікації зображень пошкоджень, використовуючи моделі нейронних мереж. Особливості побудованої системи: до неї можна звернутися з будь-якого пристрою, не вимагає завантаження додаткових модулів або бібліотек для своєї роботи, різні компоненти не залежать один від одного, тому можна розробляти/ модифікувати їх паралельно, має зрозумілий користувальницький інтерфейс, має зрозумілий програмний інтерфейс, щоб у майбутньому її можна було легко інтегрувати в іншу систему.

Розроблений програмний продукт – рішення реальної бізнес проблеми та повністю функціонуючий та повноцінний продукт. Веб додаток – це, в першу чергу, рішення страхових компаній, але може використовуватися і в персональних цілях.

Серед перспектив подальших досліджень варто відмітити дослідження та використання для розв’язання даної задачі нових, більш складних архітектур мереж, зокрема, автокодувальників, додати розпізнавання знаходження пошкоджень, навчити мережу на більшій кількості категорій, додати нові підкатегорії, опублікувати веб додаток у мережі.

Література

1. From AlexNet to NASNet: A Brief History and Introduction of Convolutional Neural Networks URL: <https://towardsdatascience.com/from-alexnet-to-nasnet-a-brief-history-and-introduction-of-convolutional-neural-networks-cf63bf3320e1>
2. Классификация. URL:  
   <http://www.machinelearning.ru/wiki/index.php?title=%D0%9A%D0%BB%D0%B0%D1%81%D1%81%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F>
3. Как сформировать датасет для машинного обучения URL: <https://www.bigdataschool.ru/blog/dataset-data-preparation.html>
4. A Comprehensive Guide to Convolutional Neural Networks URL:  
   <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>
5. Image Classification with Convolutional Neural Networks URL: <https://medium.com/@ksusorokina/image-classification-with-convolutional-neural-networks-496815db12a8#:~:text=Convolutional%20neural%20networks%20(CNN)%20is,this%20architecture%20is%20image%20classification.&text=Instead%20of%20the%20image%2C%20the%20computer%20sees%20an%20array%20of%20pixels>
6. Convolutional Neural Networks — Image Classification w. Keras URL: <https://www.learndatasci.com/tutorials/convolutional-neural-networks-image-classification/>
7. Гудфеллоу Я., Бенджио И., Курвилль А.. Глубокое обучение. пер. с анг. А. А. Слинкина. 2-е изд., испр. М.: ДМК Пресс, 2018, 652 с.
8. Various Optimization Algorithms For Training Neural Network URL: <https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6>
9. Image Classification With MobileNet URL: <https://medium.com/analytics-vidhya/image-classification-with-mobilenet-cc6fbb2cd470>
10. EfficientNet URL: <https://paperswithcode.com/method/efficientnet#:~:text=EfficientNet%20is%20a%20convolutional%20neural,resolution%20using%20a%20compound%20coefficient>
11. EfficientNet: Improving Accuracy and Efficiency through AutoML and Model Scaling URL: <https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html>
12. Convolutional Neural Network Model Innovations for Image Classification URL: <https://machinelearningmastery.com/review-of-architectural-innovations-for-convolutional-neural-networks-for-image-classification/>
13. Web application URL: <https://en.wikipedia.org/wiki/Web_application>
14. Основные понятия и особенности клиент-серверной архитектуры URL: <https://testmatick.com/ru/osnovnye-ponyatiya-i-osobennosti-klient-servernoj-arhitektury/>
15. Как работают веб приложения URL: <https://habr.com/ru/post/450282/>
16. Illustrated: 10 CNN Architectures URL:   
    <https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d>
17. Николенко С., Кадурин А., Архангельская Е. Глубокое обучение. СПб.: Питер, 2018, 480 с.