НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ КОМПЛЕКС
«ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ»
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

«На правах рукопису»
УДК 004.855.5

«До захисту допущено»
Завідувач кафедри
_____ О.Л.Тимощук
«___»_____20__ р.

Магістерська дисертація
на здобуття ступеня магістра
зі спеціальності 124 Системний аналіз
на тему: «Інтелектуальна система покращення алгоритмів стиску зображення»

Виконав (-ла):
студент (-ка) ІІ курсу, групи КА-61м
Олашин Олександр Олександрович _____
Керівник:
доцент кафедри ММСА, к.т.н., доцент
Дідковська М.В. _____
Рецензент:
доцент кафедри ПЗКС ФПМ, к.т.н., доцент
Заболотня Т. М. _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ
2018

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
НАВЧАЛЬНО-НАУКОВИЙ КОМПЛЕКС
«ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ»
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Рівень вищої освіти – другий (магістерський)

Спеціальність (спеціалізація) – 124 Системний аналіз (Системний аналіз та управління)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.Л.Тимощук
«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Олашин Олександр Олександрович

1. Тема дисертації «Інтелектуальна система покращення алгоритмів стиску зображення», науковий керівник дисертації Дідковська Марина Віталіївна, кандидат технічних наук, доцент кафедри, затверджені наказом по університету від «___»_____ 20__ р. №_____
2. Термін подання студентом дисертації _____
3. Об’єкт дослідження: системи на основі нейронних мереж для роботи з алгоритмами стиску даних
4. Предмет дослідження: алгоритми стиску зображення
5. Перелік завдань, які потрібно розробити: огляд алгоритмів та методів стиску зображення, огляд існуючих систем покращення алгоритмів стиску зображення та аналіз їх переваг та недоліків, огляд архітектур нейронних мереж та можливості їх використання для стиску даних, розробка архітектури

системи покращення алгоритмів стиску та проведення аналізу результативності розробленої системи на практиці.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: об’єкт та предмет дослідження, мета роботи, актуальність роботи, постановка задачі магістерської дисертації, аналіз існуючих методів, запропонований метод, архітектура системи, вибір платформи та мови програмування, результат роботи, результати роботи за критерієм PSNR, аналіз результатів, результати роботи за критерієм SSIM, аналіз результатів, стартап-проект «Інтеграційний пакет для MS D365 CE та MS Sharepoint», переваги, висновки.

7. Орієнтовний перелік публікацій: Олашин О. О. Система покращення алгоритмів стиску зображення // Міжнародний науковий журнал "Інтернаука". — 2018. — №9.

8. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка

Студент О.О. Олашин

Науковий керівник дисертації М.В. Дідковська

РЕФЕРАТ

Магістерська дисертація: 65 с., 11 рис., 26 табл., 3 додатки, 26 джерел.

Об’єкт дослідження: алгоритми стиску даних.

Метою даної роботи є розробка інтелектуальної системи покращення алгоритмів стиску зображень. В роботі проаналізовані концепції, підходи та методи стиску зображень, зроблений огляд існуючих робіт та систем покращення алгоритмів стиску.

Результати роботи:

– запропонована архітектура системи інтелектуального покращення алгоритмів стиску;

– реалізовано програмний продукт (REST веб сервіс) який виступає як інтерфейс системи покращення зображення;

– реалізовано стартап – проект інтеграційний пакет MS Dynamics CRM та MS Sharepoint, з використання розробленої системи покращення зображення.

Новизна роботи:

– запропоновано принципово новий спосіб побудови системи для покращення стиску зображень;

– обґрунтовано використання власних підходів до розв’язку задачі стиску зображень.

Результати даної роботи можна використовувати для побудови системи покращення стиску зображень на основі нейронних мереж. Розміщена як REST веб сервіс система доступна для використання, як частину більш складної онлайн орієнтованої системи.

НЕЙРОННА МЕРЕЖА, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, СТИСК ЗОБРАЖЕНЬ, КОМП’ЮТЕРНЕ БАЧЕННЯ, MS DYNAMICS CRM

ABSTRACT

Master thesis: 65 p., 11 figures, 26 tables, 3 annexes, 26 sources.

Object of research: algorithms of data compression

The purpose of this work is to develop an intelligent system for improving image compression algorithms. The paper analyzes concepts, approaches and methods of compression of images, reviews of existing works and systems for improving compression algorithms.

Results of work:

– the architecture of the system of intellectual improvement of compression algorithms is proposed;

– a software product (REST web service) is implemented, which serves as an interface for improving the image;

– a startup is implemented - the project of the integration package MS Dynamics CRM and MS Sharepoint, using the developed image enhancement system.

Novelty of work:

– a fundamentally new way of constructing a system for improving image compression is proposed;

– the use of own approaches to the decision of problems of compression of images is substantiated.

The results of this work can be used to build a system for improving the compression of images based on neural networks. Placed as a REST web service system is available for use as part of a more complex online-oriented system.

NEURAL NETWORK, CONVOLUTIONAL NEURAL NETWORK, IMAGE COMPRESSION, COMPUTER VISION, MS DYNAMICS CRM

ЗМІСТ

ВСТУП

Технології розвиваються надзвичайно стрімко. І їх розвиток прискорюється з кожним роком. Кількість знань людства подвоюється кожні 5 років і цей процес тільки прискорюється. Новітні технології втілюються в життя неймовірно стрімко і стають частиною нашої повсякденної рутини. Здається тільки нещодавно Інтернет почав крокувати світом, а ось вже без нього не можливо уявити собі наш освіт. І так відбувається в усіх галузях.

Однак деякі речі як не дивно залишаються доволі інертними до змін. Яскравим прикладом є алгоритми стику зображень (або ж формати зображень). Так PNG було створено в 1997, а JPEG і того раніше - в 1992 році. Здається це було нещодавно, однак по міркам сучасного технологічного зростання цілу вічність тому. Дані формати залишаються надзвичайно популярними і досі домінують на ринку зображень [1].

Однак на превеликий жаль вимоги до якості зображень з поступом технологій тільки зростають. Так якщо раніше основним критерієм до алгоритмів було стиснути зображення якомога сильніше, то зараз при покращенні дисплеїв якість зображення грає не менш важливу роль ніж його компактність. Також не варто забувати про вимоги, які накладає на зображення Інтернет – для передачі через мережу розмір зображення повинен бути досить не великим і в той же час містити достатньо інформації, щоб отримане зображення мало хорошу якість. І великі компанії як ось Google та Apple приймають кроки для створення нових алгоритмів [2][3]. Однак поки що ідеальної заміни для цих форматів не існує.

Більшість фахівців і науковців зосереджують свою увагу або на покращенні існуючих алгоритмів або на створенні абсолютно нових алгоритмів та форматів стиснення зображення. Однак на даний момент введення нових форматів, при поточному поширенні існуючих є доволі не простим завданням. Тому більш ефективним напрямком (поки не буде

створено більш революційних методів стиску), на мою думку є зосередження на методах та інструментах, що дозволять користуватись існуючими форматами файлів і в той же час покращити кінцевий результат.

Таким чином, ціллю даної роботи є створення інструменту, який не змінюючи існуючі алгоритми кодування та декодування зможе представити інструмент, який суттєво покращить якість зображень та їх розмір.

Для досягнення поставленої цілі було вирішено наступні задачі:

– проаналізовано роботи інших авторів та зроблено висновок щодо найбільш оптимального підходу для вирішення задачі;

– розроблено архітектуру системи та на її основі створено робочу модель.

Об’єктом дослідження даної роботи є системи на основі нейронних мереж для роботи з алгоритмами стиску даних

Предметом дослідження є алгоритми стиску даних.

Наукова новизна отриманих результатів полягає в наступному:

– запропоновано архітектуру системи на основі нейронних мереж для покращення вже існуючих алгоритмів стистку;

– показано ефективність запропонованої архітектури.

Практичним результатом даної роботи є розробка архітектури системи покращення алгоритмів стиску за домогою нейронних мереж, а також реалізації цієї системи.

Робота складається з 4 розділів. В першому розділі описуються та аналізуються основні алгоритми стиску зображень. Другий розді присвячений аналізу існуючих архітектур нейронних мереж. Третій розділ являє собою огляд запропонованої архітектури та демонструє результати роботи. Четвертий розділ показує стартап-проект на основі використання запропонованої системи.

1. ОГЛЯД ЗАДАЧІ СТИСКУ ЗОБРАЖЕНЬ

1.1 Аналіз предметної області

1.1.1 Аналіз алгоритмів стиску даних

Стиснення даних — це процедура перекодування даних, яка проводиться з метою зменшення їхнього обсягу, розміру, об'єму.

Стиснення даних має широке застосування в обчислювальних службах та рішеннях, зокрема, передачі даних. Стиснення даних здійснюється за допомогою декількох методів стиснення та програмних рішень, які використовують алгоритми стиснення даних для зменшення розміру даних.

Загальна методика стиснення даних вилучає і замінює повторювані елементи даних та символи, щоб зменшити розмір даних. Повторювані елементи (або інакше кажучи надлишок) усувається заміною цих послідовностей певним коротшим значенням. Стиснення даних, які не мають властивості надлишку (наприклад випадковий сигнал чи шум), неможливе. Також, зазвичай, неможливо стиснути зашифровану інформацію.

Виділяють два види стиснення даних:

– стиснення без втрат — при якому можливе відновлення вихідних даних без спотворень;

– стиснення зі втратами — при якому можливе відновлення з незначними (або в певних випадках значними) спотвореннями.

1.1.1.1 Алгоритму стиску без втрат

Стиснення без втрат передбачає стиснення даних таким чином, що оригінальний набір даних повністю реконструюється при оберненні

стиснення. Стиснення без втрат знайшло своє застосування у обробці та збереженні даних та комп'ютерних програм.

Можна привести приклади даних для яких спотворення не припустимі:

– критично важливі дані, будь яка змінах в яких може призвести до жахливих наслідків: наприклад, дані отримані медичними апаратами або приладами контролю та управління (в літаках, потягах, космічних апаратах та іншого транспорту), тощо;

– будь які текстові/символічні дані, що залежать від семантики так контексту, зміна в яких змінює його – наприклад комп’ютерні програми (включно з їх вихідними файлами) , виконавчі масиви тощо;

– дані, що проходили процедуру стиснення неодноразово у процесі їх обробки (наприклад при створенні відео, музики та багато іншого), тощо.

Наведемо приклади найбільш відомих алгоритмів стиску без втрат.

Алгоритм RLE (Run Length Encoding) — один з найдавніших та найлегших алгоритмів стиску графіки. Зображення в ньому витягується в ланцюжок байт по рядках растра. Якщо пояснювати простіше то алгоритм замінює однакові послідовності символів на відповідні їм пари типу кількість/значення. Наприклад рядок «АААААББББГГГ» перетвориться у щось накшталт «4А4Б3Г». Основний напрямок застосування алгоритму це зображення з невеликою кількістю кольорів: наприклад професійні та наукові зображення. Однак не зважаючи на свою простоту, швидкість роботи та надзвичайно низьке використання пам’яті при процесі арахівації та деархівації він не знайшов широкого застосування через не високу ефективність[4].

Алгоритм LZW – названий в честь його розробників Lempel, Ziv і Welch. Основою стиснення в даному алгоритм, на противагу RLE, слугують однакові ланцюжки байтів. Алгоритм стиснення можна подати наступним чином – спочатку послідовно зчитуємо всі символи зі входу і перевіряємо, чи існує в таблиці рядків (яку ми створюємо) такий рядок. Якщо збіг знайдено, то ми продовжуємо процес читання (по символьно), якщо ж такого рядку немає, то

рядок заноситься в таблицю і процес повторюється. Основним цільовими зображеннями даного алгоритму є 8-бітові зображення. Варто зазначити високу універсальність даного алгоритму – його різноманітні варіанти знайшли застосування у звичайних архіваторах [4].

Алгоритм Хаффмана — вважається один з класичних алгоритмів, які відомі доволі давно (ще з 60-х). У своїй роботі опирається тільки на частоту появи однакових байт в зображенні. Алгоритм зіставляє символам з вхідного потоку, що трапляються найчастіше, ланцюжок біт меншої довжини. І, навпаки, тим, які зустрічається достатньо рідко — відповідно ланцюжок більшої довжини. У своїй роботі вимагає дворазового проходження по вихідному зображенні. Варто зазначити, що алгоритм Хаффмана не застосовують як самодостатній алгоритм. На практиці він являє собою частину схеми більш складних і багатоетапних алгоритмів стиснення. Прикладом цього може слугувати формат даних TIFF, де Хаффман лише один з декількох кроків. До переваг алгоритму варто віднести простоту реалізації, швидкість роботи та можливість апаратної реалізації [4].

Алгоритм Lossless JPEG був створений групою експертів в області фотографії в 1993 році. Даний алгоритм являє собою спеціальну версію JPEG без втрат. Основна цільова група Lossless JPEG являє собою повноколірні зображення (24-бітні) або зображення в градаціях сірого без палітри (8-бітові). Основна цільова аудиторія для застосування Lossless JPEG необхідність існування побітової відповідності між вихідним та стиснутим зображенням (наприклад медичні зображення). Однак алгоритм так і не отримав широкого розповсюдження [4].

Алгоритм JBIG створений експертами Joint Bi-level Experts Group у 1993 році. Його призначення – стиск бінарних чорно-білих зображень (в основному чорно-білих сканів так факсів). Його також можна застосовувати для більш складних зображень (2-х та 4-х бітових). У випадку більш складних зображень алгоритм розділяє їх на окремі бітові площини. До переваг алгоритму JBIG можна віднести можливість керування певними параметрами результуючого

зображення (порядком розбиття, шириною смуг та рівнями масштабування). Варто зазначити, що однією з причин чому алгоритм не набув популярності були проблеми з патентуванням [4].

Підсумовуючи, можна сказати, що алгоритми стиснення без втрат можуть бути застосовані до більш широкого колу даних та є загалом більш універсальними.

Однак не варто забувати, що стиснення без втрат має свої обмеження, а саме існують дані, які не тільки не можна стиснути, але й стиснений варіант яких буде займати більше місця ніж вихідне зображення.

1.1.1.2 Алгоритми стиску з втратами

Основне поле застосування стиснення зі втратами для зменшення розмірів відео, зображень, музики, бо для такого типу інформації це виявилось надзвичайно вигідним, оскільки людина зазвичай не помічає не великих артефактів. Однак потрібно бути обережним з обсягами застосування цих алгоритмів – більш інтенсивне використання

Стиснення з втратами — метод стиснення даних, при якому файл, що пройшов стиснення може відрізнятись від вихідного образу, однак достатньо інформативний для використання. Цей тип стиснення найбільш поширений у зв’язку зі стисненням мультимедійних даних (відео, зображень, музики) та при передачі даних через інтернет. В цьому контексті такі методи часто називаються кодеками.

Кодеки можна поділити на дві категорії:

– трансформуючі кодеки – програма використовує знання про об’єкт кодування, розрізають на невеликі частини, при цьому лишня інформація відкидається. Отриманий результат стискають різноманітними

методами. Фінальний файл може відрізнятись від вихідного, однак достатньо схожий на нього, що дозволяє подальше використання;

– предиктові кодеки – використовують попередні та/або наступні дані для того, щоб передбачати поточний елемент (будь то зображення чи звук). Помилка між отриманим таким чином елементом та справжніми даними, разом з іншою інформацією необхідною для передбачення квантується та кодується.

В деяких існуючих системах застосовується комбінація цих двох кодеків через використання трансформуючих кодеків для стиску помилкових сигналів, згенерованих предиктовим кодеком на етапі передбачення.

Якщо порівняти методи стиску з втратами з методами стиску без втрат не важко побачити, що у випадках коли не потрібно дотримуватись побітової ідентичності вихідного і кінцевого файлу методи стиску з втратами перемагають через суттєво більші ступені стиску, оскільки в більшості випадків людські органи почуттів не помічають змін. Однак варто зазначити, що при збільшені ступеню стиску зростає імовірність утворення більш помітних змін.

Зважаючи на те, що більшості дослідників чудово відомо про ці особливості людських органів, вони будували свої моделі спираючись на ці «недоліки». Зміни, що виникають у результуючих файлах називаються артефактами стику.

Приклади алгоритмів з втратами:

JPEG (Joint Photographic Experts Group) — растровий формат збереження графічної інформації, що використовує стиснення з втратами. Набув надзвичайно широкого застосування і перетворився практично на стандарт.

Алгоритм стиснення даних, що використовується у форматі, базується на алгоритмі дискретного косинусного перетворення. Алгоритм можна розділити на декілька етапів:

– підготовка;

– дискретне косинусне перетворення;

– квантування;

– повторний стиск.

Формат JPEG здобув широке розповсюдження і на даний момент має надзвичайно велике поширення серед комп'ютерних браузерів[1] (рисунок 1.1).

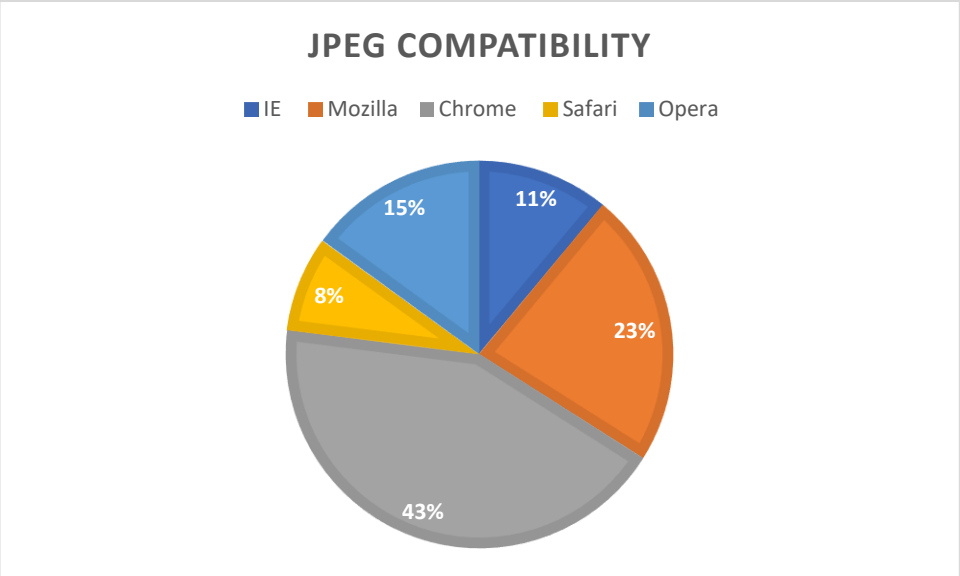


Рисунок 1.1 – Підтримка JPEG між різними версіями браузерів

Найбільша роздільна здатність, яку підтримує формат JPEG/JFIF є 65535×65535.

Варто зауважити, що на відмінну від PNG JPEG не підтримує прозорість, а на відміну від GIF не підтримує анімацію.

Фрактальне стиснення зображень —це математичний процес для кодування растрів, які містять реальне зображення, в сукупність математичних даних, що описують фрактальні властивості зображення. Цей вид кодування заснований на тому, що усі природні і більшість штучних об'єктів містять надмірну інформацію у вигляді однакових малюнків, що повторюються[4]. Найбільш відомим цей алгоритм став в зв'язку з тим, що при певних випадках можна отримати неймовірно високий коефіцієнт стиснення (до 1000 разів) і

при тому зберегти прийнятний вигляд зображення. Однак цей алгоритм не отримав широкого застосування – сказались патентні проблеми.

Рекурсивний алгоритм (або хвильовий) – базується на використанні когернтних областей. Використовується для зображень з плавними переходами (як для кольорових так і для чорно білих). Можливо досягнути стиснення да 100 разів включно. Найбільш ефективно підходить до картинок по типу рентгенівських знімків.

1.2 Аналіз існуючих підходів до покращення алгоритмів стиску

На даний момент існує багато різноманітних підходів, які обирають дослідники для покращення алгоритмів стиску. Одні обирають шлях створення нових алгоритмів (наприклад Google та Apple[1]). Інші йдуть шляхом оптимізації та покращення вже існуючих алгоритмів. Zhai у свої роботі [5] запропонував ефективний метод для JPEG зображень, що базується на пост-фільтрації зображення для зниження шумів, що з’явились після компресії. Foi [6] у своїй роботі пропонує використання фільтрів на основі пристосування форм зображення. Також варто зазначити роботу Zhang [7], чий алгоритм фільтрації має надзвичайно ефективні результати і в той же час має доволі доступний спосіб реалізації.

Досягнення в області нейронних мереж та їх широке поширення вплинуло і на роботи по оптимізації роботи алгоритмів стиску зображення. Так використання згорткових мереж та рекурентних мереж в задачах комп’ютерного зору [8][9][10] показало їх неймовірну ефективність та вказало шлях для інших дослідників.

Серед дослідників, які працюють над задачею стиску даних в сфері нейронних мереж можна виділити два найбільші напрямки. Перший з них це напрямок який займається покращенням роздільної здатності зображень

(image super resolution). Цей напрямок характеризує створення нейронних мереж, які суттєво покращують фінальне зображення. Dong [11] запропонував метод покращення роздільної здатності зображення на основні згорткової нейронної мережі, названий ним SRCNN, яка складається з трьох шарів. Однак сам автор статті вважає, що глибокі нейронні мережі призводять до суттєвого покращення результатів, дослідження інших авторів показують суттєвий приріст. Так мережа DRCN [12] має 16 рекурсивних шарів і результати її роботи, перевершують результати аналогічних алгоритмів доволі суттєво.

Іншим шляхом, які обрали дослідники є методи стиску зображень що базуються безпосередньо на роботі нейронних мереж. Наприклад Toderici [13] у свої роботі запропонував фреймворк на основі згорткових та розгорткових нейронних мереж. Більш того в своїх наступних роботах [14] цей автор пропонує ще більш ефективний підхід, який показує відмінні результати при різноманітних розмірах зображень та типах стиску.

З найбільш нещодавніх робіт варто відмітити роботу Suman Kunwar [15]. В цій роботі автор пропонує алгоритм використання згорткової нейронної мережі для реалізації стиснення даних у форматі JPEG. До переваг запропонованого алгоритму є простота реалізації та швидкодія роботи фінальної мережі.

1.3 Постановка задачі

Аналіз існуючи методів стиску зображень та робіт інших авторів показує, що більшість з них намагаються або покращити вже існуючі методі стиску або запропонувати абсолютно нові методи.

Таким чином основним завданням даної роботи є розробка архітектури системи покращення алгоритмів стиску за домогою нейронних мереж та реалізації даної системи з практичної перевіркою її ефективності.

Висновки за розділом

Алгоритми стиснення зображень мають широке застосування та розповсюдження. Всі існуючі алгоритми стиску зображень можна поділити на дві основні частини – алгоритми стиску з втратами та алгоритми стиску без втрат.

Назви цих двох сімейств алгоритмів повністю їх характеризують та окреслюють область застосування: алгоритми стиску без втрати необхідні для збереження інформації, яка не повинна бути зміненою чи пошкодженою (наприклад для наукового чи медичного застосування). З іншого боку алгоритми стиску з втратами розповсюджено використовуються в тих областях, де існування невеликих артефактів на зображень не є проблемою.

Серед алгоритмів стиску з втратами варто виділити JPEG та PNG, як алгоритми з найбільшим застосуванням. Саме ці алгоритми були обранні, для перевірки створеної системи в зв'язку зі своїм широким застосуванням.

Не зважаючи на широке різноманіття можливих інструментів, які можна застосувати для розв’язку задачі стиску зображень більшість дослідників зосередились на двох основних напрямках – створення нових алгоритмів стиску або використання нейронних мереж для покращення вже існуючих. Зважаючи на широке розповсюдження нейронних мереж та успіхи, яких досягнули інші дослідники використовуючи цей інструмент, було вирішено зосередити увагу на нейронних мережах, як на основному засобі для вирішення поставленої задачі.

2. АНАЛІЗ НЕЙРОННИХ МЕРЕЖ

2.1 Аналіз нейронних мереж

Штучні нейронні мережі (ШНМ) — це обчислювальні системи, в основу яких лягли біологічні нейронні мережі, що утворюють мозок людей та тварин. Перевагою таких систем є можливість навчання для вирішення задач (вони поступово покращують результати) на основі прикладів, без попереднього програмування під конкретну задачу. Так наприклад, у задачі розпізнавання зображень нейронні мережі можуть навчитися розпізнавати зображення, що містять людей, спираючись на аналіз зображень-прикладів, які розмічені як «людина» і «не людина». Мережі роблять це не маючи жодного апріорного поняття про людину, їх визначні характеристики (волосся, голова, два уха тощо), тощо. Мережа сама розробляє свій власний набір характеристик з наданих прикладів.

ШНМ ґрунтується на створенні сукупності з'єднаних вузлів, які мають назву штучні нейрони (проводячи аналогію з біологічними нейронами в мізках тварин та людей). Кожне з'єднання (аналогічне синапсам) між штучними нейронами може передавати сигнал від одного до іншого. Штучний нейрон, що отримує сигнал, може обробляти його, і потім сигналізувати штучним нейронам, приєднаним до нього.

В найбільш розповсюджених реалізаціях нейронних мереж сигнал на з'єднанні між штучними нейронами являє собою дійсне число, а вихід кожного штучного нейрону обчислюється нелінійною функцією суми його входів. Штучні нейрони та з'єднання зазвичай мають вагу, яка підлаштовується в перебігу навчання. Вага збільшує або зменшує силу сигналу на з'єднанні. Штучні нейрони можуть мати такий поріг, що сигнал надсилається лише якщо сукупний сигнал перетинає цей поріг. Штучні нейрони зазвичай організовано в шари. Різні шари можуть виконувати різні види перетворень своїх входів.

Сигнали проходять від першого (входового) до останнього (вихідного) шару, і під час руху можуть проходити означені шари декілька разів.

Спочатку штучні нейронні мережі розроблялись з метою, навчитись розв’язувати поставлені задачі таким же чином, як це робить людський мозок. Однак з часом дослідники відхилились від аналогів з біологічними системи і почали створення більш штучних архітектур, спеціалізованих під певні задачі. На даний момент нейронні мережі застосовуються для вирішення широкого кола задач включаючи розпізнавання природньої мови, комп’ютерне бачення, машинний переклад, аналіз тональності, грою в шахи, грою в комп’ютерні ігри та діагностика медичних захворювань.

2.1.1 Рекурентні нейронні мережі

Рекурентні нейронні мережі — це один з класів штучних нейронних мереж, у якому з'єднання між нейронами утворюють орієнтований цикл. Дана особливість створює внутрішній стан мережі, який в свою чергу надає системі змогу динамічно змінюватись з часом. На відміну від нейронних мереж прямого поширення, рекурентні нейронні мережі мають змогу використовувати свою внутрішню пам'ять для обробки довільних послідовностей входів. Саме ця особливість робить їх досить ефективними в задачах розпізнавання мовлення та рукописного вводу.

Повнорекурентна мережа — це основна архітектура, яка була створена в 1980-х роках: мережа нейроноподібних вузлів, кожен з яких з’єднаний з кожним іншим (з’єднання орієнтоване). Кожен з нейронів має змінну в часі дійснозначну активацію. Кожне з'єднання має змінювану вагу (також дійсну). Певний набір нейронів на які подаються вхідні дані називаються вхідними нейронами, набір нейронів, що відповідає за вихід називається вихідними, а нейрони в яких відбуваються основні обчислення називаються прихованими.

Для постановок керованого навчання з дискретним часом тренувальні послідовності входових векторів стають послідовностями активацій входових вузлів, по одному вектору на кожен момент часу. В будь-який момент часу кожне не вхідний нейрон обчислює свою поточну функцію активації (зазвичай це нелінійна функція від зваженої суми функцій активації всіх з’єднаних з ним (орієнтовано) вузлів). Також, якщо відбувається навчання з учителем на певних тактах навчання на вихідні нейрони може подаватись додаткові активації. В кожен заданий момент часу кожен не входовий вузол обчислює свою поточну активацію як нелінійну функцію від зваженої суми активацій всіх вузлів, від яких до нього надходять з'єднання.

Якщо розглядати випадок навчання з прикріпленням, то в ньому не існує вчителя, що може надавати сигнали активації. Замість нього використовується функція пристосованості або інакше кажучи функція винагороди, що оцінює продуктивність рекурентної нейронної мережі.

Рекурсивна нейронна мережа створюється при рекурсивному застосуванні одного й того ж набору ваг до диференційовної графоподібної структури шляхом обходу цієї структури в топологічній послідовності. Також, в загальному такі мережі тренують методом зворотного поширення похибки. Рекурсивні нейронні мережі отримали широке застосування в задачах розпізнавання природньої мови та розпізнавань зображень.

Яскравим прикладом таких нейронних мереж є мережа Хопфілда.

2.1.1.1 Мережа Хопфілда

Нейронна мережа Хопфілда – являє собою рекурентну, повнозв’язну нейронну мережу з симетричною матрицею зв’язків. Особливістю цього типу мереж є те, що динаміка цих мереж сходиться до одного з можливих положень

рівноваги. Такі положення рівноваги являють собою локальний мінімум функціоналу, який має назву енергія мережі.

Приклад архітектури мережі Хопфілда представлено на рисунку 2.1

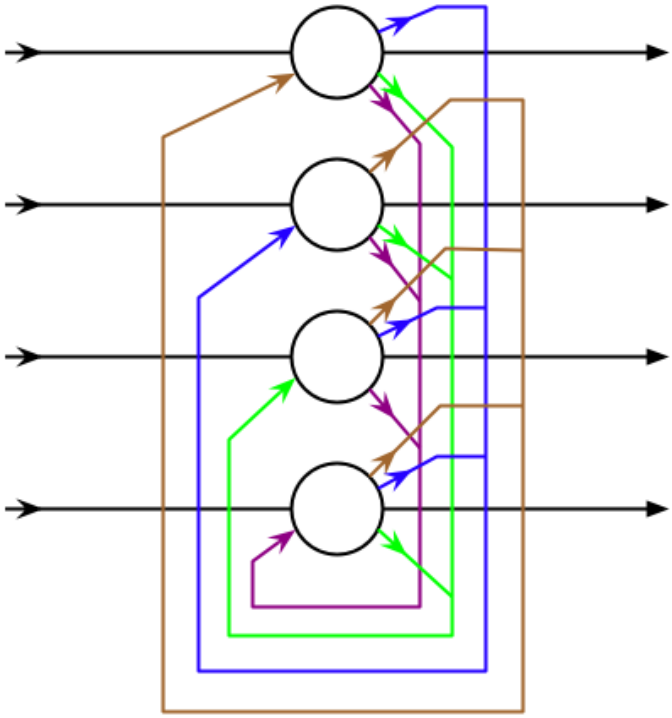


Рисунок 2.1 – Архітектура мережі Хопфілда з 4ма нейронами

Також необхідно зазначити особливість даної мережі в плані тренування – всі коефіцієнти матриці вагів розраховуються однією формулою та за один цикл, на відміну від стандартних методів навчання (наприклад метод оберненого розповсюдження похибки), де ваги коригуються ітеративно.

Даний тип нейронних мереж набув широкого розповсюдження в зв’язку з можливістю використання даної мережі як автоасоціативної пам’яті та для розв’язку задач оптимізації.

Яскравим прикладом застосування даної мережі є відновлення зашумлених простих зображень. Спочатку мережу тренують з використанням набору вихідних зображень і після того як тренування закінчене отриману мережу можна застосувати для відновлення зображень (рисунок 2.2).

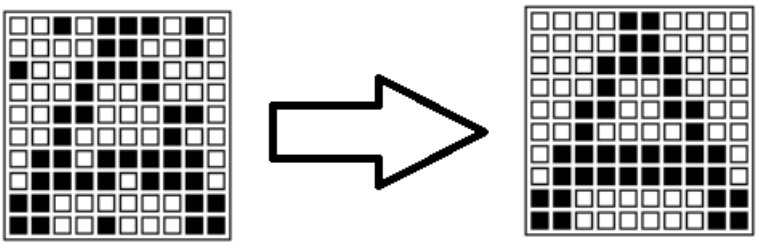


Рисунок 2.2 – Відновлення зображення за допомогою мережі Хопфілда

Однак мережа Хопфілда має свої недоліки:

– відносно не великий об’єм пам’яті доступний для запам’ятовування;

– досягнення рівноваги не гарантує отримання правильної відповіді.

Через наведені вище недоліки застосування даної мережі до більших за розміром зображень (а також зображень, які мають велику кількість класів для розпізнавання).

2.1.2 Згорткові нейронні мережі

Згорткові нейронні мережі — це клас глибинних штучних нейронних мереж прямого поширення, які знайшли своє застосування в задач розпізнавання та аналізу зображень.

Згорткові нейронні мережі базуються на використанні різновиду багатошарових перцептронів, та розроблені таким чином, щоби вимагати використання мінімального обсягу попередньої обробки зображень. Інші розповсюджена назва згорткових нейронних мереж – мережі інваріантні відносно зсуву або просторово інваріантні штучні нейронні мережі. Ця назва походить від їх архітектури спільних ваг та характеристик інваріантності відносно паралельного перенесення.

Дослідники, які створили згорткові нейронні мережі черпали натхнення в біологічних процесах мозку людина, а архітектура цих мереж базується на організації зорової кори людини.. Окремі нейрони кори реагують на стимули лише в обмеженій області зорового поля, відомій як рецептивне поле. Рецептивні поля різних нейронів частково перекриваються таким чином, що вони покривають усе зорове поле. Саме це надає згортковим нейроним мережам досягати високих результатів в задачах аналізу зображень.

Згорткові нейронні мережі використовують відносно невелику кількість попередньої обробки, в порівнянні з іншими алгоритмами класифікації зображень. Це означає, що мережа сама створює фільтри, які в інших алгоритмах задаються дослідниками власноруч. Ця незалежність від вчителя є однією з найбільш суттєвих переваг цього типу нейронних мереж перед іншим, адже дозволяє суттєво зекономити в плані ресурсів.

Основне поле застосування цих нейронних мереж лежить в задача комп’ютерного зору, розпізнавання зображень, їх класифікації, а також в завдання розпізнавання природньої мови. Саме в цих задачах згорткові нейронні мережі демонструють найкращі результати.

Стандартна архітектура згорткових нейронних мереж містить шари входу та виходу, а також певну кількість прихованих шарів між ними. Ці шари можуть бути різноманітними (наприклад шари агрегування, повноз’єднанні шари, шари нормалізації). Приклад архітектури можна спостерігати на рисунку 2.3.

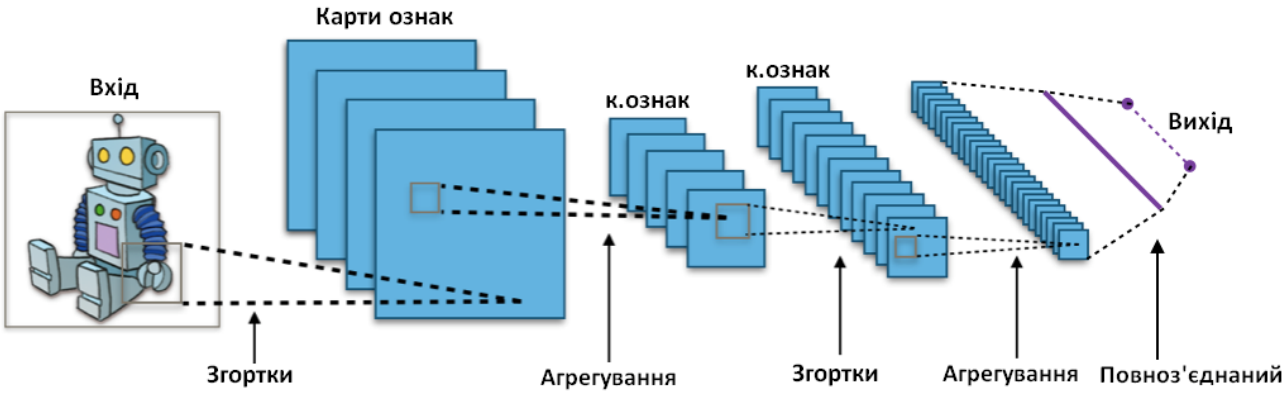


Рисунок 2.3 – Архітектура згорткової нейронної мережі

Згорткові шари застосовують до вхідних даних операцію згортки, і передають отриманий результат до наступного шару. Операція згортки імітує реакцію окремого нейрону зорової кори на зоровий стимул.

Важливо зауважити, що кожен окремий згортковий нейрон обробляє дані лише для свого рецептивного поля.

Незважаючи на те, що можливе використання повноз’єднаних нейронних мереж прямого поширення для задач комп’ютерного бачення та розпізнавання зображенньь ефективність даного рішення є надзвичайно низькою. Через особливості архітектури було б необхідним використання надзвичайно велкиого числа нейронів, оскільки у випадку зображень розмір входу був би надзвичайно великим, через те, що кожен піксель зображення був би окремою змінною. Приведемо приклад. Нехай ми маємо повноз’єднаний шар для відносно маленького зображення розміром 100 × 100. Такий шар буде містити 10 000 ваг. Саме тут в нагоді стають згорткові мережі, оскільки операція згортки дозволяє суттєво скоротити розміри мережі, дозволяючи збільшувати її глибину, зменшуючи при цьому кількість параметрів.

Згорткові нейронні мережі, як частину своєї архітектури можу мати шари агрегування (як локального так і глобального), які об’єднують виходи груп нейронів одного шару до одного з нейронів наступного шару. Є різні види агрегування. Так, наприклад, одним з найбільш вживаних є максимізаційне агрегування – беремо максимальне значення з кожної з групи нейронів

попереднього шару. Іншим доволі популярним типом агрегування є усереднювальне агрегування – обраховане середнж значення кожної з груп нейронів поточного шару передається до наступного [16].

На відміну від стандартної архітектури повноз’єднаних мереж, де кожен нейрон поточного шару з’єднаний з кожним нейроном наступного шару (наприклд багатошаровий перцептрон), згорткові нейронні мережі використовують спільні ваги, тобто, до кожного рецептивного поля шару застосовується один фільтр. Як результат обсяг пам’яті, необхідної для виконання операцій зменшується і фінальна продуктивнність зростає.

2.2 Порівняння та вибір архітектури нейронної мережі

Рекурентні мережі незважаючи на свою популярність в різноманітних задач і широке застосування в задач розпізнавання зображень в останні роки почали суттєво програвати згортковим нейронним мережам в цій області. Застосування згорткових нейронних мереж розв'язує проблему зникання або вибухового збільшення градієнтів у тренуванні традиційних багатошарових нейронних мереж з багатьма шарами за допомогою зворотного поширення. Це в свою чергу дозволяє створювати більш глибокі мережі, що в свою чергу покращує кінцевий результат.

Незважаючи на велике використання рекурентних мереж в задачах розпізнавання зображень, згорткові мережі мають найбільш широке та ефективне застосування, коли йдеться про визначення основних структурних характеристик зображень, що дозволяє застосовувати згорткові нейронні мережі для створення компактних версій вихідних зображень.

Саме через означенні вище причини та характеристики згорткова нейронна мережа була обрана, як основний інструмент для реалізації поставленої задачі.

2.3 Аналіз архітектури для системи покращення алгоритмів стиску

Оскільки, для створення системи було обрано згорткові нейронні мережі, як найбільш перспективні з огляду задачі розпізнавання зображень, варто більш детально розглянути особливості будови нейронних мереж для необхідної системи.

Більш детально архітектуру системи буде розглянуто в розділі 3. Достатньо зауважити, що в архітектурі буде застосовано дві системи – перша для створення зменшеної версії вхідного зображення. Друга для покращення для відновлення вихідного зображення. З огляду на завдання поставлені перед данними мережами й буде розглянуто архітектурні особливості кожної з них.

2.3.1 Мережа для створення зменшеної версії зображення

Отже нам потрібно побудувати нейронну мережу, яка може створити зменшенну версію зображення і при цьому зберегти всі основні структурні елементи зображення. Це робиться для того щоб вибраний для системи кодер міг працювати максимально ефектино на зменшеній копії зображення.

Для того щоб створити зменшену копії зображення нам знодобиться згорткова нейронна мережа з декількома шарами.

Очевидно, що перший згортковий шар нейронної мережі буде використано для того щоб отримати основні характеристики зображення. Однак варто вибрати кількість фільтрів, а також функції активації для даного шару.

В якості функції активації було обрано ReLU (рисунок 2.4)

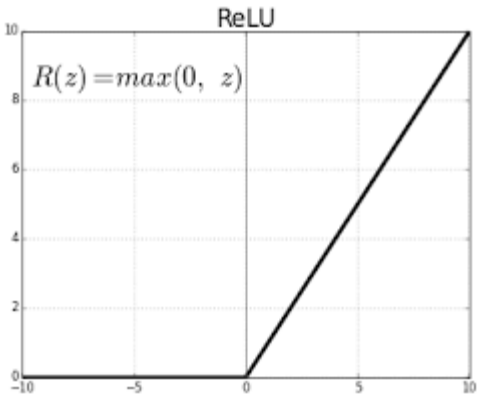


Рисунок 2.4 – ReLU

Примітка. Рисунок 2.4 взяту з курсу CS With James [17]

Дану функцію було обрано, як функцію активації є те, що більшість дослідників в своїх працях на рахунок згорткових нейронних мереж зазначають, що вона найкраще підходить для задачі розпізнавання зображень.

Вибір кількості шарів та фільтрів залежить від складності задачі та від наявних ресурсів. Чим більше шарів та фільтрів, тим більше деталей можна виявити та зменшити зображення. Однак збільшення кількості, як шарів так і фільтрів призводить до збвльшення час тренування та використовуваних ресурсів. Таблиця 2.1 показує залежність між кількістю шарів, часом тренування та точністю вихідного зображення (за критерій було обрано SSIM – індекс структурної схожості).

Таблиця 2.1 – Кількість шарів, фільтрі, час тренування та точність вихідного зображення

К-ть шарів	К-ть фільтрів	Час тренування	SSIM
2	20	2	0.5
2	40	5	0.76
2	50	7	0.78
2	60	10	0.81

Продовження таблиці 2.1

К-ть шарів	К-ть фільтрів	Час тренування	SSIM
3	20	3	0.6
3	40	6	0.86
3	50	8	0.92
3	60	12	0.93
4	50	14	0.935
5	50	19	0.94

Як можна побачити з таблиці наведеної вижче результати при кількості шарів більше 3 і кількості фільтрів більше 50 сутєєво не зростають, а ось час необхідний для тренування зростає суттєво. Тому використововувати системи з більшою кількості шарів можливо й принесло б кращі результати, однак не надто відмінні від тих, які можна отримати при використанні 3 шарів та 50 фільтрів.

Варто зазначити, що в практичній реалізації було використано 48 фільтрів. Це пов’язано з технічними обмеження обраної платформи на час тренування. Відмінність між 48 і 50 фільтрами майже не суттєва, однак тільки 48 фільтрів вклалися в технічні обмеження системи.

2.3.2 Мережа для відновлення вихідного зображення

Мережа для відновлення вихідного зображення повинна бути достатньо ефективною, щоб не тільки відновити отримане зображення, але й по можливості покращити його.

Зрозуміло, що кількість фільтрів, які будуть використані в цій мережі, дорівняю 48 – саме стільки фільтрів ми застосовуємо в першій мережі. Однак кількість шарів не настільки очевидна річ. Як показали роботи інших дослідників, наприклад [11] навіть 3 шарів достатньо для того, щоб відновити і покращити фінальне зображення. Однак, як показують інші роботи збільшення шарів дозволяє отримати набагато кращі результати [12]. Більшість авторів погоджується, що 20 шарів на данному етапі дають найкращи результат. Однак через технічні особливості використовуваної для практичної реалізації платформи було обрано 18 шарів.

Висновки за розділом

Нейронні мережі отримали широке поширення і набуваються все більшої популярності. Існує надзвичайно велика кількість різноманітних архітектур та підходів до нейромереж, наприклад рекурентні мережі чи згорткові нейронні мережі.

Обидва види нейронних мереж отримали широке застосування в задачах пов’язаних з стиском та розпізнаванням зображень. Так Google Reasearch Group у свої статті [18] чудово описало застування рекуретної нейронної мережі для задачі стиску зображення і привела приклади, що ілюструють ефективність даної мережі.

Однак в більш нових роботах все більше і більше дослідників використовують згорткову нейронну мережу, як більш ефективний інструмент.

Застосування згорткових нейронних мереж розв'язує проблему зникання або вибухового збільшення градієнтів у тренуванні традиційних багатошарових нейронних мереж з багатьма шарами за допомогою зворотного

поширення. Це в свою чергу дозволяє створювати більш глибокі мережі, що в свою чергу покращує кінцевий результат.

Наведенні вище причини, а також персональна зацікавленність в згорткових нейронних мережах стали основними аргументами на користь використання згорткової нейронної мережі як основи для розв’язку поставленої задачі.

Також було запропоновано архітектуру мереж застосовуваних в системі: мережі для зменшення зображення та мережі для відновлення вихідного зображення.

3. АНАЛІЗ АРХІТЕКТУРИ ПРОГРАМНОГО ПРОДУКТУ

3.1 Обґрунтування вибору платформи та мови програмування

На поточний момент на ринку представлено неймовірне різноманіття фреймворків, систем та засобів для створення нейронних мереж. Їх можна написати на багатьох мовах програмування або навіть не застосовувати жодної мови програмування.

Однак при побудові нейронної мережі не варто забувати, що однією з найважчих в плані використання ресурсів є частина тренування або навчання нейронної мережі. Так багато архітектур нейронних мереж дозволяють розпаралелити обчислення і використовувати для навчання потужності не тільки центрального процесора, а й відеокарти. Але все одно навчання достатньо складної нейронної мережі може займати надзвичайно тривалий час.

До того ж після тренування нейронної мережі постає інша задача – експортувати отриману нейронну мережу для її наступного використання. Звичайно використання таких фреймворків, як Caffe2 дозволяє відносно легко розгортати отриману мережу, однак ми все одно не можемо говорити про універсальність.

Багато компаній намагаються запропонувати власні системи та інструменти для створення нейронних мереж. Яскравим прикладом може слугувати Google AI (попередньо Google ML)[19], Amazon Machine Learning [20] та Azure Machine Learning Studio [21]. Кожна з названих служб надає інструменти та платформу, які можуть бути використані для побудови надзвичайно складних та ефективних нейронних мереж. Однак кожна з них має свої обмеження, як от розмір доступного місця не сервері, вартість тренування нейронної мережі (по кількості операцій, по кількості задіяних ресурсів, по кількості годин навантаження ресурсу) та багато інших. Порівняння даних сервісів наведено в таблиці 3.1.

Таблиця 3.1 – Порівняння сервісів машинного навчання

	Amazon ML	Google AI	Azure ML
Класифікація	+	+	+
Регресія	+	+	+
Кластеризація	-	+	+
Визначення аномалій	-	+	+
Рекомендації	-	+	+
Ранжування	-	+	+
Алгоритми	невідомо	TensorFlow-based	100+
Підтримка фреймворків	-	TensorFlow	-
Наявність графічного інтерфейсу	-	-	+
Рівень автоматизації	Високий	Низький	Середній

Звичайно найлегшим шляхом було б написання нейронної мережі без використання наведених вище сервісів з використанням Python чи R, однак зусилля, які знадобилися б на тренування та розміщення такого сервісу не покрили б переваги легкості написання.

Серед сервісів Azure ML Studio виділяється декількома суттєвими перевагами – безкоштовність використання (в певних межах звісно ж), можливістю використання вже натренованих мереж, баз даних та прикладів (в інших сервісах теж є така можливість, однак їх галереї суттєво менші), можливість використання готових блоків для створення/візуалізації нейронної мережі та й проекту загалом. До того ж однією з ключових переваг Azure ML Studio стало можливість легкого розміщення натренованої мережі як REST веб

ресурсу прямо в Azure Web Storage без додаткових затрат (одним натиском кнопки).

До мінусів вибраного сервісу варто віднести потребу у використанні мови програмування Net# - це мова програмування, яку Microsoft розробила спеціально, як засіб створення та опису нейронних мереж. Не зважаючи на доволі простий синтаксис, вона має багато специфічних речей, які варто мати на увазі перед побудовою нейронної мережі. І не зважаючи на те що ми можемо писати модулі на Python та R та включати їх в проект, на даний момент всі нейронні мережі доступні для застосування є або пре створеними Microsoft мережами або повинні бути написані на Net#.

Не зважаючи на існування певних мінусів Azure Machine Learning Studio було обрано, як основне середовище для створення та розміщення нейронної мережі. Як мови програмування було обрано Net# та Python.

3.2 Аналіз архітектури програмного продукту

3.2.1 Система покращення алгоритмів стиску

Компресія зображень традиційно була одним з завдань, в якому нейронні мережі повинні були бути надзвичайно хорошими однак нажаль не було достатньо доказів які підтверджували б можливість створення нейронної мережі яка б ефективно працювала б для компресії зображень з різними алгоритмами та різними розмірами зображень. [22] показали, що можна навчати єдину рекурентну нейронну мережу і досягти кращих, ніж найсучасніші, показники стиснення для заданої якості незалежно від вхідного зображення, але автори обмежувались зображеннями розміром 32 × 32. Мета цієї роботи – створення системи нейронних мереж, яка забезпечує ефективні результати при використанні різних алгоритмів та різних розмірів зображень. Є три можливі способи досягти цього:

– спроектувати потужний кодер/декодер на основі різних покращенн базових алгоритмів;

– розробка універсального кодеру, який здатний фіксувати довгострокові залежності між патернами на зображенн;

– використання існуючих кодерів разом з системою нейронних мереж для покращення процесу стиску та відновлення зображення.

У даній роботі я вибираю третій варіант, як найбільш перспективний та ефектний в плані кінцевого результату. Для того, щоб оцінити, наскільки добре запропоновані архітектури (тобто "якість"), в даній роботі використовуються два загальновживані показники – коефіцієнт пікового сигналу до шумового співвідношення (PSNR) та індекс структурної схожості (SSIM).

Варто зазначити, що для людського зору відмінності в показаннях цих показників не завжди є повністю очевидними. Це відбуваєтсья оскільки людська зорова система є більш чутливою до певних видів спотворень, ніж іншію Ця ідея використовувалася в методах стиснення зображень з втратою, таких як JPEG. Щоб бути в змозі виміряти такі відмінності, потрібно застосовувати засіб для спостереження за візуальною системою людини, який в ідеалі має співвідноситись із тим, як люди сприймають відмінності зображення.

На жаль, в літературі існує велика різноманітність показників різної якості, більшість з яких не використовуються тільки в певних роботах та не є загальновживаними. В інших роботах автоенкодери були використані для зменшення розмірності зображень, перетворення зображень у стиснуті двійкові коди для пошуку та отримання компактних візуальних уявлень, які можуть бути використані в інших програмах. Зовсім недавно, варіаційні (рекурентні) автоенкодери безпосередньо застосовувалися до проблеми стиснення (з результатами на зображеннях розміром до 64 × 64 пікселів), а для варіативного кодування використовувалися неваріаційні періодичні нейронні

мережі. У більшості нейронних мереж стиснення зображень використовується фіксований ступінь стиснення залежно від розміру шару вузьких місць [16].

Основою підходу запропонового в цій роботі є ідея використання нейронних мереж не як основи процесу стиснення, а як інструменту покращення вже існуючих алгоритмів. Таким чином досягається використання форматів, які надзвичайно поширення і можливість розширення можливостей сторонніх алгоритмів стиску за допомогою згорткової нейронної мережі. Архітектура складається з подвійної мережі. Перша мережа, яка буде приймати зображення і генерувати компактне представлення (Compact CNN, надалі CCNN). Вихід цієї мережі буде оброблений стандартним кодеком (наприклад, JPEG). Пройшовши через кодек, зображення буде передане до другої мережі, яка буде "виправляти" зображення з кодека, намагаючись повернути оригінальне зображення - Reconstructive CNN (RCNN). Обидві мережі ітеративно навчаються та схожі на GAN (Generative adversarial network – генеративні змагальні мережі).

На рисунку зображеному нижче (рисунок 3.1) можна побачити архітектуру запропонованої системи.

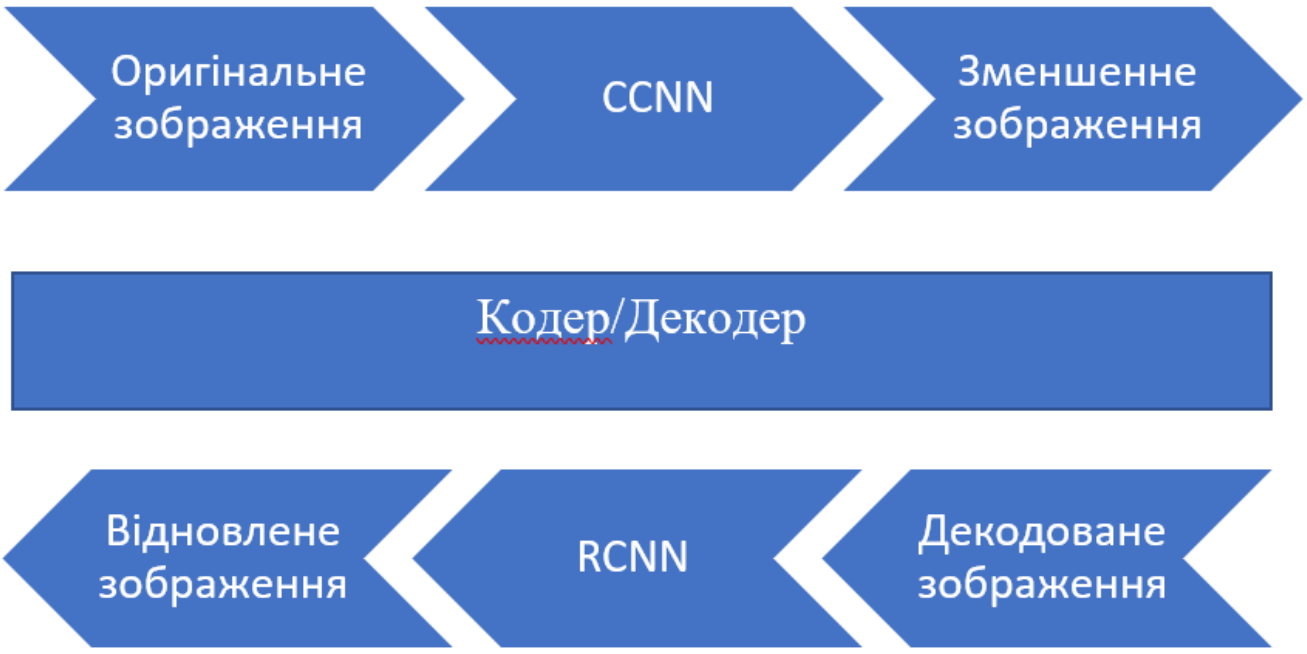


Рисунок 3.1 – Архітектура запропонованої системи

Вихід із кодека збільшується, а потім передається в RCNN. RCNN спробує вивести зображення, яке буде виглядати максимально схоже на оригінальне зображення (в деяких випадках навіть краще ніж оригінальне зображення).

CCNN складається з 3х шарів, які дозволяють зберегти основну структуру вхідного зображення і як результат дозволяють кодеку ефективніше стискати зображення.

Перший шар являє собою згортковий шар, що використовує ReLU, як функцію активації. Цей шар використовується для того, щоб отримати основні характеристики зображення. Фільтри мають розмір 3×3×c, де c – кількість каналів вихідного зображення. В даній роботі використовується 48 фільтрів (через обмеження обчислювальних потужностей). Нелійність досягається використанням ReLU як функції активації. Наступний шар виконує стандартну функцію для більшості згорткових нейронних мереж – зменшення зображення та покращення отриманих характеристик. Для цього застосовується операція згортки з кроком, що дорівнює два. Фільтри даного шару мають розміри 3×3×48 і ReLU також застосовується як функція активації. Для останнього третього шару використовують c фільтрів розміру 3×3×48 для створення компактного представлення зображення.

RCNN складається з 18 шарів. Перший шар складається з 48 фільтрів розміром 3×3×c і використовується для створення 48 карт характеристик. Як функція активації використовується ReLU. Потім йдуть 16 шарів використовуються 48 фільтрів розміром 3×3×48. Варто зазначити, що в кожному з цих шарів також використовується нормалізація між операціями згортки та ReLU. Для останнього шару використовується c фільтрів розміром 3×3×48 для відновлення зображення.

Залишок можна розглядати як етап після обробки, щоб «покращити» зображення, яке декодував кодек. Нейронна мережа, яка має багато «інформації» про світ, може приймати когнітивні рішення про те, що «виправити». Ця ідея базується на залишковому навчанні.

Оскільки існує дві мережі, використовуються дві функції втрат. Перша, для CCNN, позначена як L1, визначається за формулою (3.1):

$$L_1(\theta_1) = \frac{1}{2N}\sum_{k=1}^{N}\left\|Re\left(\hat{\theta}_2, Cr(\theta_1, x_k)\right) - x_k\right\|^2 \quad (3.1)$$

де N – кількість нейронів;

$\hat{\theta}_2$ – тренувальні параметри RCNN;

θ_1 – тренувальні параметри CCNN;

x_k – вхідне зображення.

$$Cr(\theta_1, x_k), \quad (3.2)$$

$$Re\left(\hat{\theta}_2, Cr(\theta_1, x_k)\right) \quad (3.3)$$

Cr (3.2) позначає вихід CCNN. Re (3.3) позначає RCNN. Це рівняння просто передає значення рівняння 1 до RCNN. Шапка θ позначає тренувальні параметри RCNN (шапка означає, що параметри фіксуються).

Рівняння (3.1) зробить CCNN модифікованими його вагами такими, що після відтворення RCNN, остаточне зображення виглядатиме якнайближче до реального зображення вводу.

Друга функція втрат для RCNN визначається як:

$$L_2(\theta_2) = \frac{1}{2N}\sum_{k=1}^{N}\left\|res\left(Co(\hat{x}_{m_k}), \theta_2\right) - \left(Co\left(\hat{x}_{m_k}\right) - x_k\right)\right\|^2 \quad (3.4)$$

де Co () позначає вихід з кодека;

$\hat{x}$ позначає вихід з CCNN;

θ_2 позначає тренувальні параметри RCNN.

Варто зазначити, що RCNN навчається на різницю між Co () і вхідним зображенням, а не безпосередньо з вхідного зображення.

Рівняння (3.4) змусить RCNN змінювати свої ваги таким чином, щоб вихідний файл виглядав як можна ближче до оригінального зображення.

Моделі навчаються ітеративно, подібно до того, як тренуються GAN. Одна вага моделі фіксується, коли оновлюються інші ваги моделі, потім інші ваги моделі фіксуються, тоді як перша модель тренується.

3.2.2. Архітектура мережі в середовищі Azure ML Studio

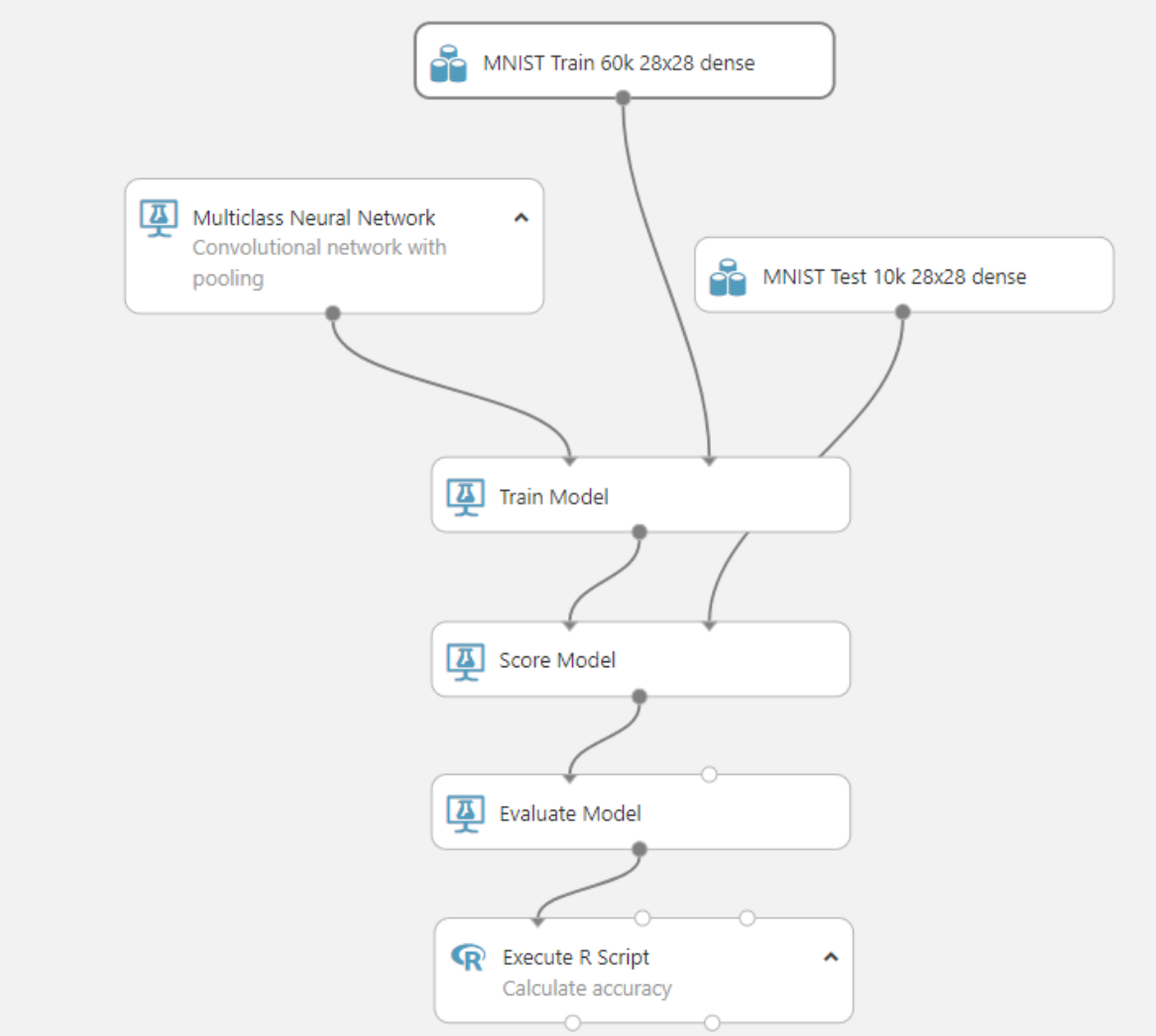


Рисунок 3.2 – Архітектура нейронної мережі в Azure ML Studio

На рисунку наведеному вище (рисунок 3.2) можна побачити спрощену репрезентацію архітектури нейронної мережі в Azure ML Studio. Наведене зображення відображає архітектуру CCNN. Як основу для даної архітектури було взято шаблон наданий Microsoft для побудови простої згорткової нейронної мережі (на рисунку Multiclass Neural Network). Не велике пояснення: в Azure ML Studio не існує окремого об’єкту, який репрезентує згорткову нейронну мережу. На даний момент доступними є Two Class Neural Network та Multiclass Neural Network в різних варіаціях. Для того щоб отримати згорткову нейронну мережу необхідно обрати Multiclass Neural Network та змінити її внутрішній код за допомогою Net#.

3.3 Аналіз результатів роботи

Для тренування було використано 300 зображень розміром 180×180 пікселів. Тренування зайняло 24 години.

Варто зазначити, що можливе покращення отриманих результатів (що будуть наведені нижче) збільшити кількість зображень використаних для тренування.

Для порівняння було обрано два основні показники – коефіцієнт пікового сигналу до шумового співвідношення (PSNR) та індекс структурної схожості (SSIM).

Дані роботи системи порівнювались зі стандартним алгоритмом стиску JPEG, так і з двома робототами які вважаються надзвичайно ефективними у своєму виконанні (Zhangs та ARCNN).

Для тестування використовувались 5 класичних зображень, які застосовуються для тестування. Вони наведенні на рисунку 3.3.

Рисунок 3.3 – Тестові зображення

Результати роботи наведені в таблицях 3.2-3.3 та на рисунках 3.4 та 3.5.

Таблиця 3.2 – Результати роботи за критерієм PSNR (dB)

Метод	Метелик	Дім	Папуга	Листя	Фотограф	Середній результат
JPEG	22.58	27.77	26.19	22.49	24.45	20.18
Zhang	24.20	29.24	27.78	24.13	25.39	21.308
ARCNN	25.64	29.68	28.13	25.07	25.27	21.63
CCNN	22.85	27.9	27.01	22.85	24.78	20.508
RCNN	24.55	28.34	27.57	24.23	25.17	21.062
Система	25.78	29.78	28.01	25.03	25.58	21.68

Таблиця 3.3 – Результати роботи за критерієм SSIM

Метод	Метелик	Дім	Папуга	Листя	Фотограф	Середній результат
JPEG	0.7378	0.7733	0.7581	0.7775	0.7283	0.60744
Zhang	0.8313	0.8141	0.8308	0.8548	0.7672	0.65338
ARCNN	0.8741	0.8209	0.8446	0.8983	0.7674	0.66624
CCNN	0.7201	0.8019	0.8232	0.7761	0.7582	0.63188
RCNN	0.867	0.8189	0.8411	0.8643	0.7543	0.65572
Система	0.8732	0.8321	0.8367	0.8821	0.7677	0.66372

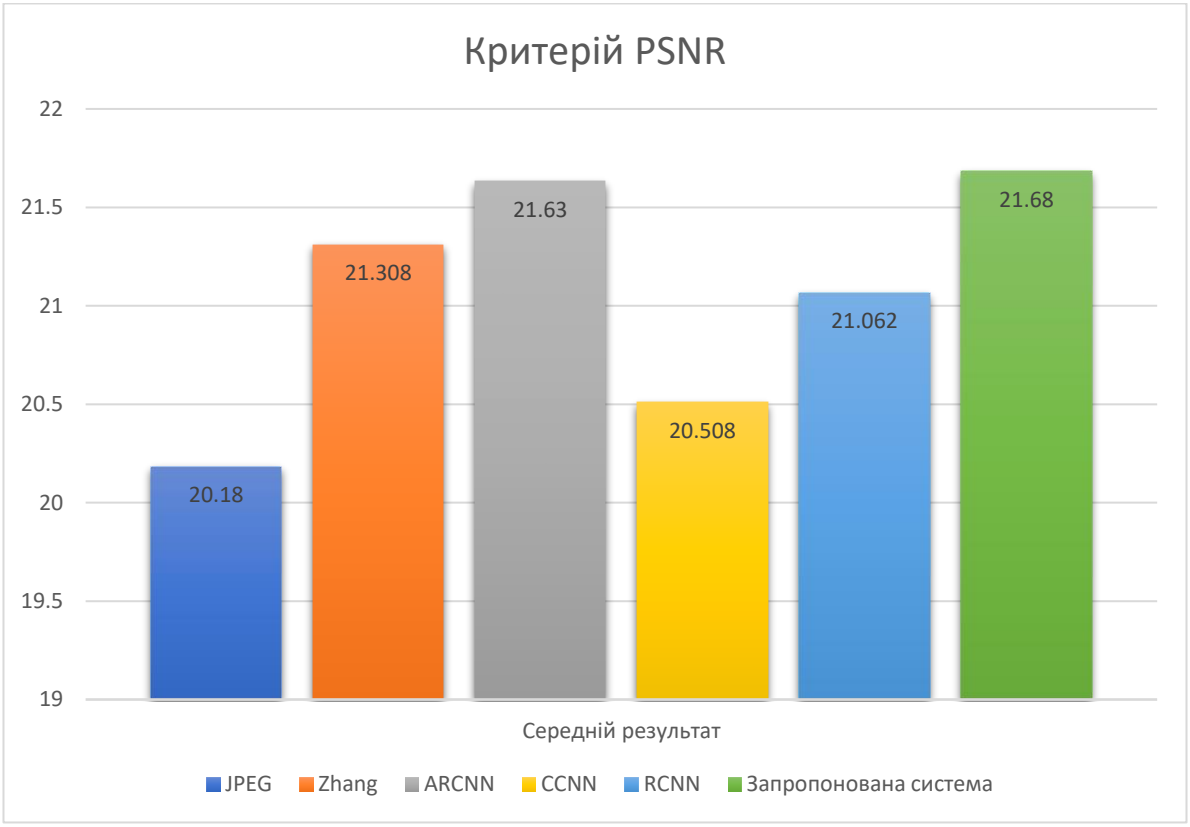


Рисунок 3.4 – Середній результат за критерієм PSNR

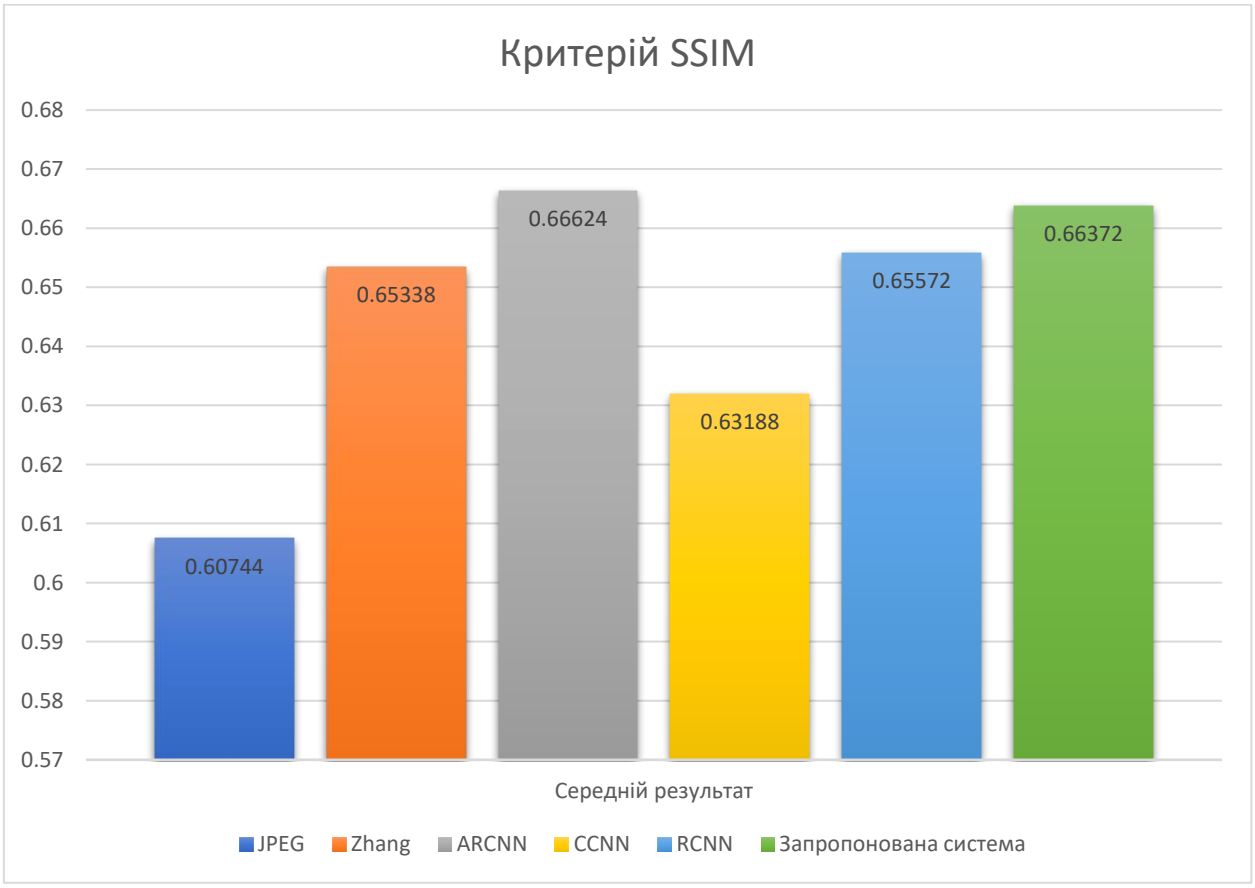


Рисунок 3.5 – Середній результат за критерієм SSIM

Таблиця містить порівняння з іншими популярними алгоритмами стиску зображень, так і результати кожної мережі окремо.

За критерієм PSNR запропонована система показала найвищий середній результат 21.68, перегнавши ARCNN, результат якої - 21.63. За критерієм же SSIM розроблена мережа показали другий середній результат споміж протестованих систем (0.66372 проти 0.66624 ARCNN). Середній результат було обрано як основний, тому що можна помітити, що на деяких зображеннях система показувала результат найвищий серед інших, а на деяких трохи відставала від ARCNN.

Отримані результати без сумніву можна вважати надзвичайно хорошими. ARCNN є однією з найкращих систем, створених на даний момент, і те що отримана система змогла не тільки наблизитись до результатів цієї мережі, а й перевершити їх є дуже хорошим досягненням.

Щодо стандартного алгоритму JPEG – не важко побачити, що запропонована система надзвичайно перевершує стандартний алгоритм. Варто зазначити, що саме стандартни JPEG було використано, як кодер/декодер для тестування, що показує ефективність використання системи зі стандартними алгоритмами.

Висновки за розділом

Серед різноманітних підходів до вирішення задачі стиску зображень, в цій роботі було обрано той, який в перспективі повинен дати найбільшу універсальність. Так використання нейронних мереж, не як основного інструменту для стиску зображення, а як допоміжного елементу в системі зі стандартним алгоритмом стиску довело свою ефектиність. Результати таблиць 3.2 та 3.3 показують, що фінальний результат роботи системи є надзвичайно якісним – сердні показник PSNR дорівнював 21.68, що більше аналогічного

показнику однієї з найкращих мереж на даний момент ARCNN (21.63), а результат SSIM 0.66372 лише трохи відстав від ARCNN (0.66624)

Наведений в цій роботі алгоритм може мати надзвичайно широке застосування – починаючи від різних хостингів збереження зображень, закінчуючи персональним використанням.

До переваг запропонованого інструменту слід віднести покращену якість зображень та зменшення розміру результуючого зображення. Ці характеристики роблять його ідеальним інструментом для використання у мережі Інтернет. Також слід виділити його універсальність – він може бути застосований з багатьма існуючими алгоритмами.

Також варто зазначити, що розміщення натренованої мережі, як веб ресурсу на Azure дозволяє легко використовувати його, як елемент будь-якої складної системи. До того ж висока стабільність Azure дозволяє говорити про високу надійність та доступність сервісу 24/7.

4. РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

4.1 Опис ідеї проекту

На даний момент CRM система створена компанією Microsoft є однією з найбільших CRM систем на ринку [23]. Вона має як свої переваги так і певні недоліки (як нажаль будь-яке інше програмне забезпечення на ринку). Одним з найбільших недоліків системи на даний момент є доволі не зручна система інтеграції з іншими пов'язаними продуктами Microsoft. Яскравим прикладом такої проблеми є інтеграція з MS Sharepoint – хмарна система збереження документів та роботи з ними, яка насправді може набагато більше[24].

Проблемо MS Dynamics CRM Online є дороговизна збільшення пам'яті. Для того щоб вирішити цю проблему (а також розширити асортимент програмного забезпечення, яке використовує клієнт) Microsoft включає в стандартну ліцензію можливість використання MS Sharepoint. Однак не зручність використання, не велика кількість тренувальних матеріалів та не можливість перегляду файлів з середини CRM системи роблять цей варіант далеко не найбільш оптимальним для користувачів.

Зважаючи на все сказане вище не дивно, що на ринку існують більш оптимальні інтеграційні пакети від сторонніх розробників, які базуються на стандартній інтеграції між MS Dynamics CRM та MS Sharepoint. Однак всі вони мають свої недоліки, які вирішуються в запропонованому стартап-проекті (таблиці 4.1 – 4.22).

Таблиця 4.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Створення пакету інтеграції	Використання для збереження документів та зображень	Дозволяє надзвичайно сильно зменшити витрати на

Продовження таблиці 4.1

Зміст ідеї	Напрямки застосування	Вигоди для користувача
MS Dynamics CRM та MS Sharepoint		розширення доступного місця в MS Dynamics CRM та надає зручний інтерфейс для роботи з документами та зображеннями в рамках однієї системи.

Таблиця 4.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів				W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	PowerObjcet	Magnetic One	Initial integration			
1.	Швидкість роботи	+	+	+	+			+
2.	Легкість використання	+	+	+/-	-			+
3.	Стиск даних	+	-	-	-			+
4.	Ціна	+	-	-	+/-			+

4.2 Технологічний аудит ідеї проекту

Таблиця 4.3 – Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
		Система нейронних мереж	Розробити	Розроблено автором проекту

Продовження таблиці 4.3

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
	для стиску зображень	Інтеграція між MS Dynamics CRM та MS Sharepoint	Розробити	Розроблено автором проекту
		.NET CRM package	Наявна	Доступна
		OData Endpoint	Наявна	Доступна
Обрана технологія реалізації ідеї проекту: Для реалізації проекту було обрано Згорткові нейронні мережі як основу системи нейронних мереж для стиску зображень та розроблено інтеграційний пакет.				

Запропонований проект має архітектуру (спрощений вигляд) вказану на рисунку 4.1.

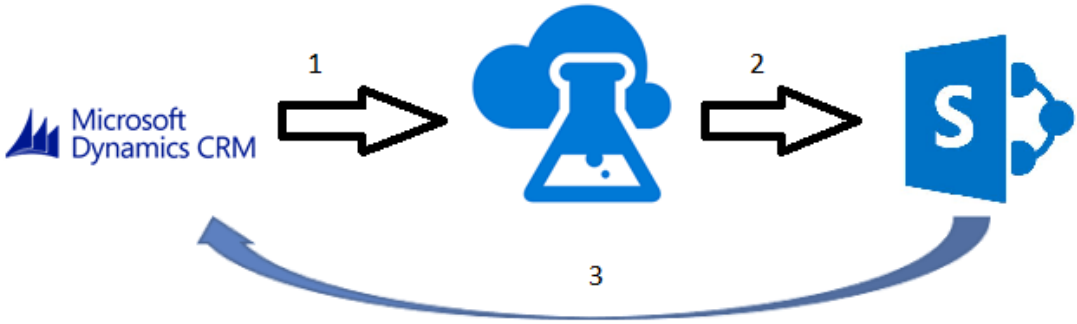


Рисунок 4.1 – Архітектура проекту

Він складається з трьох складових:

а) файл для стиску надсилається з MS Dynamics CRM до Azure ML де стискається;

б) файл надсилається до MS Sharepoint де й зберігається;

в) посилання на файл відправляється назад до MS Dynamics CRM.

4.3 Аналіз ринкових можливостей запуску стартап-проекту

Таблиця 4.4 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	4
2	Загальний обсяг продаж, грн/ум.од	
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі (або по ринку), %	10%

Таблиця 4.5 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
	Можливість використання файлів в MS Dynamics CRM без збільшення коштів на їх зберігання в системі	Компанії які застосовують MS Dynamics CRM, що у свої роботі повинні застосовувати великий об’єм файлів або зображень як частину своєї щоденної роботи	Легкість та зрозумілість користування	- швидкість роботи та легкість в користуванні - підтримка продукту, тренуваннння з користування

Таблиця 4.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
	Зміна онлайн сховища	Зміна Sharepoint на інше онлайн сховище як Google Drive чи Dropbox	Створення підтримки для сторінніх сховищ та підтримки функції єдиної авторизації
	Створення стандартного інтеграційного пакету	Створення компанією Microsoft стандартного програмного пакету інтеграції	Розширення можливого функціоналу пакету (додаткові можливості). Створення підтримки для сторонніх онлайн сховищ

Таблиця 4.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
	Збільшення кількості форматів файлів	Збільшення кількості форматів файлів, які використовують для роботи компанії	Розширення функціоналу для редагування та обробки більшої кількості форматів (включаючи відео та аудіо формати)
	Збільшення кількості файлів	Збільшення об’єму файлів для щоденного використання	Використання алгоритмів стиску даних для збільшення кількості файлів для збереження

Таблиця 4.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Вказати тип конкуренції - монополістична	На потенційних клієнтів впливає не тільки ціна продукту, але й його функціонал	Підтримка відносно низької ціни при високій якості програмного продукту
2. За рівнем конкурентної боротьби - міжнародний	Всі компанії оперують на міжнародному ринку	Реалізація підтримки декількох мов в програмному середовищі та тех підтримки на декількох мовах

Продовження таблиці 4.8

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
3. За галузевою ознакою - внутрішньогалузева	Існуючі інтеграційні пакети зосередженні тільки на одному програмному продукті	Реалзіувати підтримку попередніх версій MS Dynamics CRM (з 2013 версії) та сторонніх хмарних середовищ
4. Конкуренція за видами товарів: - товарно-видова	Багато схожиш інтеграційних пакетів	Створення функцій, які суттєво виділяють власний продукт на іншими
5. За характером конкурентних переваг - нецінова	Створення нових функціональних особливостей відбувається надзвичайно стрімко	Постійне вдосконалення та роширення основоного функціоналу продукту
6. За інтенсивністю - марочна	Основні компанії, які давно зарекомендували себе відіграють основну роль	Створення бренду та підтримка його репутації, як основна парадигма

Таблиця 4.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	PowerObjects MsCrmAddons	Існує багато потенційних	Основними факторами сили	Сила споживачі	Створення стандартного

Продовження таблиці 4.9

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	ProAddon Magnetic One	конкурентів, однак єдиним бар'єрами входження на ринок є створення хоча б мінімального функціоналу	постачальників є широкий функціонал та відомий бренд	тільки в оцінці зручності роботи з ПП	інтеграційного пакету
Висновки:	Найбільшу долю ринку займають PowerObjects та MsCrmAddons	Можливість виходу на ринок є надзвичайно простою та привабливою з точки зору прикладених зусиль. Не зважаючи на можливу кількість потенційних конкурентів шанси на успіх при правильному функціоналі дуже хороші.	Єдине,що диктують найбільші конкуренти є мінімальний функціональний рівень програмного продукту	Клієнти не мають визначальної долі на ринку.	Основною загрозую є існування можливості створення компанією Microsoft стандартного повнофункціонального інтеграційного пакету

Таблиця 4.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Зрозумілий та зручний інтерфейс	Більшість користувачів є звичайними користувачами комп'ютеру і для них є

Продовження таблиці 4.10

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
		надзвичайно важливо мати легкий та зручний інтерфейс
2	Широкий функціонал	Більшість програмних продуктів конкурентів можуть мати інтеграцію між платформами, однак більшість на цьому й зупиняються. Розширення можливостей може стати вирішальним для клієнта
3	Використання алгоритмів стиску	Основне завдання пакету – збереження місця всередині MS Dynamics CRM. Однак місце на Sharepoint також скінченне і хоча розширити його доволі дешево, використання алгоритмів стиску даних робить продукт ще більш привабливим.

Таблиця 4.11 – Порівняльний аналіз сильних та слабких сторін «назва проекту»

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з … (назва підприємства)						
			–3	*–2*	*–1*	*0*	*+1*	*+2*	*+3*
1	Зрозумілий та зручний інтерфейс	15				+			
2	Широкий функціонал	12					+		
3	Використання алгоритмів стиску	18	+						

Таблиця 4.12 – SWOT- аналіз стартап-проекту

Сильні сторони: Використання алгоритмів стиску Зрозумілий та зручний функціонал	Слабкі сторони: Функціонал
Можливості: Збільшення кількості підтримуваних файлів	Загрози: Зміна онлайн-сховища Створення стандартного інтеграційного пакету

Таблиця 4.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
	Додавання кількості підтримуванних файлів	Середня	2 місяці
	Зміна онлайн-сховища	Висока	2 місяці
	Розширення стандартного функціоналу (робота з файлами)	Висока	3 місяці

4.4 Розроблення ринкової стратегії проекту

Таблиця 4.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Компанії які застосовують MS Dynamics CRM, і застосовують великий об’єм файлів або зображень як частину своєї щоденної роботи	Висока	Високий	Середня	Проста (реалізація стандартного функціоналу)
Які цільові групи обрано: існує єдина цільова група (надзвичайно широка)					

Таблиця 4.15 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
	Розширення функціональних можливостей програмного проекту	Створення та розвиток функціоналу який не доступний конкурентам	Застосування алгоритмів стиску даних та підтримки інших хмарних-сховищ	Стратегія диференціації

Таблиця 4.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
	Ні	Змішана стратегія – залучення нових клієнтів та перехоплення клієнтів конкурентів	Так, а саме – зручний інтерфейс (покращений порівняно з іншими) та функціонал (розширений згідно стратегії розвитку)	Стратегія наслідування лідеру

Таблиця 4.17 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
	Легкість та зрозумілість користування	Стратегія диференціації	Алгоритм стиску даних як основа збереження коштів, швидкість роботи та менеджмент файлів	Найдійнсть, швидкість, функціональність

4.5 Розроблення маркетингової програми стартап-проекту

Таблиця 4.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
	Збереження файлів на сторонній платформі	Забезпечує потребу	Стиснення файлів без втрати якості, для суттєвого збереження місця, швидкість роботи, широкий функціонал для роботи з файлами, низька ціна

Таблиця 4.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Автоматичне збереження файлів на сторонній платформі (Sharepoint) та можливість роботи з ними всередині MS Dynamics CRM		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Швидкість 2. Простота використання 3. Стиснення даних	Нм Нм Нм	Тх/Тл Тх/Тл/Е Тх/Тл
	Не існує стандартних нормативів. Використання автоматизованих тестів		
	Пакування: Managed Solution for MS Dynamics CRM		

Продовження таблиці 4.19

	Марка: StarLine Star365 Attachments
III. Товар із підкріпленням	До продажу: розширення для MS Dynamics CRM для автоматичного збереження файлів на MS Sharepoint та можливість робити з ними всередині MS Dynamics CRM
	Після продажу: швидкодія, постійне розширення функціоналу
За рахунок чого потенційний товар буде захищено від копіювання: фінальний продукт буде програмним продуктом з закритим кодом, тому скопіювати його не вдасться однак завадити конкурентам використовувати власні ідеї немає можливості.	

Таблиця 4.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
	1-2 $/per user	1-2 $/per user	> 100k $	0.25 $ - 2 $

Таблиця 4.21 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
	Клієнт завантажує продукт з сайту компанії	-	-	-

Таблиця 4.22 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
	Компанії, що користуються MS Dynamics CRM	Професійні соц-мережі	Легкість та функціональність	Розповісти про переваги додатку	Легкість та інноваційність на сторожі вашого рахунку

4.6 Архітектура програного продукту

Реалізацію запропонованого стартап проекту можна розділи на три основні частини:

- інтерфейс для роботи в MS Dynamics CRM;
- плагін для синхронізації MS Dynamics CRM та MS Sharepoint;
- система нейронних мереж розміщена в Azure Cloud.

Система нейронних мереж являє собою описану в розділі 3 систему стиску. Її архітектура та особливості були описані в розділі 3.

4.6.1 Інтерфейс для роботи в MS Dynamics CRM

Інтерфейс для роботи з файлами в MS Dynamics CRM складається з наступних основних складових:

- html веб ресурс, який відображає зображення у вигляді мініатюр та повноформатної галереї;
- js файл, який реалізує всю функціональну частину наведеного вище веб ресурсу.

Html сторінка також надає можливості додавання та видалення файлів з середини середовища MS Dynamics CRM. Однак потрібно зауважити декілька важливих моментів – на даному етапі можливий тільки перегляд зображень (передбачається додавання можливості видалення зображень з MS Sharepoint) та їх завантаження.

Висновки за розділом

Основна мета даного стартап-проекту – надати користувачам системи MS Dynamics CRM можливість легкої роботи з документами та зображеннями всередині системи та можливість серйозно заощадити кошти реалізуючи інтеграційний пакет з MS SharePoint.

Існує велика можливість ринкової комерціалізації за рахунок високого попиту та швидкої динаміки розвитку ринку. Варто також зазначити швидкий розвиток впровадження MS Dynamics CRM серед великих компаній та появу необхідності в додаткових інтеграційних пакетах.

Стан конкуренції на ринку на даний момент доволі задовільний – існує декілька основних гравців, які однак не показують найвищу можливу якість продукту. Поріг входження на ринок опосердковий, тому конкуренція в ніші ситуаційно-цільових рішень доволі висока, однак якщо розглядати повноцінні інтеграційні рішення отримуємо протилежну ситуацію – загалом тільки декілька великих гравців займається такою реалізацією.

Альтернативою впровадження даного інтеграційного проекту можна вважати використання технології стиску даних як окремого сервісу, який не залежить від конкретної системи чи інтеграційного пакету. Це дозволить розширити ринок використання, а можливість доступу до системи через певну систему ключових точок дозволить її використання, як складову сторонніх застосунків, що дозволяє вийти ще й на ринок мобільних та десктопних застосунків.

Подальша імплементація стартап-проекту є доцільною та рентабельною. Доцільним є також розширення через реалізацію технології стиску як окремого онлайн-сервісу і побудови на його основі інших технологічних пакетів, або ж просто використання його як посередника для інших гравців на ринку. Доцільним є також подальший розвиток технології, як однієї з найбільших конкурентних переваг на ринку.

ВИСНОВКИ

Технології вже давно стала невід’ємною частиною нашого життя. Вони постійно покращуються й модифікуються і з ними змінюється наша життя. Однак деякі речі змінюються повільніше за інші. Яскравим прикладом цього є алгоритми стиску зображень. Не зважаючи на їх широку розповсюдженість ми використовуємо алгоритми, які вже не зовсім відповідають нашим потребам.

Алгоритми стику з втратами набули стрімкого розвитку в 90ті роки. Більшість з них були створенні з ціллю зберегти достатню якість зображення при мінімізації їх розміру. Однак з покращення швидкості Інтернету та екранів недоліки алгоритмів проступають все сильніше і сильніше. Найяскравіше це можна продемонструвати на прикладі JPEG – артефакти на зображеннях високої контрастності з’являються вже через декілька циклів стиску. І хоча цей формат є надзвичайно популярним [1] його заміна це лише питання часу.

Створення нових алгоритмів стиску зображень є задачею з якою не просто впоратись. Великі компанії такі як Google та Apple зараз знаходяться на передньому краї досліджень і вже мають декілька проривних алгоритмів.[2][3] Однак впровадження нових форматів довгий і кропіткий процес. Тому метою цієї роботи було створення способів покращення алгоритмів стиску без зміни самих алгоритмів.

Для досягнення мети основним інструментом було обрано нейронні мережі. Вони набувають надзвичайну популярність і застосовуються зараз майже всюди. Однак чомусь до поставленої задачі зазвичай був єдиний підхід – натренувати мережу на одному алгоритмі і покращити лице його. І це шлях який веде в нікуди – він не є універсальним і не може бути широко застосованим. В цій роботі був запропонований інший шлях – використати дві нейромережі для пре обробки зображення для максимізації корисної дії алгоритму стиску. Було створено дві згорткові нейронні мережі (які є оптимальними в питаннях роботи з зображеннями та глибокими мережами).

Результати роботи програми були продемонстровані раніше. Результуюча комбінова система виявилась кращою за існуючі дослідницькі аналоги, а також кращою за деякі комерційно застосовані мережі [25].

До переваг запропонованого інструменту слід віднести покращену якість зображень та зменшення розміру результуючого зображення. Ці характеристики роблять його ідеальним інструментом для використання у мережі Інтернет. Також слід виділити його універсальність – він може бути застосований з багатьма існуючими алгоритмами.

До слабких сторін – потребу тренування мережі та її початкової побудови. Однак дана проблема може бути вирішена використання різноманітних бібліотек створення нейромереж (наприклад Caffe2 [26]).

Подальший розвиток даного інструменту та подальші дослідження можуть привести навіть до більш суттєвих результатів.

ПЕРЕЛІК ПОСИЛАНЬ

1. JPEG, PNG, or GIF? The Ultimate Cheat Sheet of Image File Formats [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Jami Oetting – 28.07.2017 – Режим доступу: https://blog.hubspot.com/agency/image-file-formats-infographic
2. Your photos just had a massive change with iOS 11. Here's what happened [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Stan Horaczek – 17.09.2017 – Режим доступу: https://www.popsci.com/HEIF-iphone-photos-ios-11
3. Google’s Royalty-Free Answer to HEVC: A Look at AV1 and the Future of Video Codecs [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Steven Zimmerman – 15.03.2017 – Режим доступу: https://web.archive.org/web/20170614042710/https://www.xda-developers.com/av1-future-video-codecs-google-hevc/
4. FileFormat Info [Електронний ресурс]: [Веб-сайт]. – Електронні дані. Режим доступу: https://www.fileformat.info/mirror/egff/ch09_03.htm
5. Efficient image deblocking based on postfiltering in shifted windows / [G. Zhai, W. Zhang, X. Yang et al] // IEEE Transactions on Circuits and Systems for Video Technology – 2008. – Vol.18. – PP. 122– 126.
6. A. Foi. Pointwise shape-adaptive dct for high-quality denoising and deblocking of grayscale and color images / A. Foi, V. Katkovnik, K. Egiazarian. // IEEE Transactions on Image Processing – 2007. – Vol. 16. – PP. 1395–1411.
7. R. Zhang. Image postprocessing by non-local kuans filter / R. Zhang, W. Ouyang, W.-K. Cham. // Journal of Visual Communication and Image Representation – 2011. – Vol. 22. – PP. 251–262.

8. ImageNet Classification with Deep Convolutional Neural Networks [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton - 2012 – Режим доступу: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks
9. MatConvNet: Convolutional Neural Networks for MATLAB MM '15 [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Andrea Vedaldi, Karel Lenc. - 2015– Режим доступу: https://dl.acm.org/citation.cfm?id=2807412
10. Multi-column deep neural networks for image classification Computer Vision and Pattern Recognition (CVPR) [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Dan Ciregan, Ueli Meier, Jürgen Schmidhuber. – 2012 – Режим доступу: http://people.idsia.ch/~juergen/cvpr2012.pdf
11. Image super-resolution using deep convolutional networks / [C. Dong, C. C. Loy, K. He, X. Tang.]. // IEEE transactions on pattern analysis and machine intelligence – 2016. – Vol. 38. – PP. 295–307.
12. Deeply-recursive convolutional network for image superresolution [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Jiwon Kim, Jung Kwon Lee, Kyoung Mu Lee. – 2015 – Режим доступу: https://arxiv.org/abs/1511.04491
13. Variable rate image compression with recurrent neural networks [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, R. Sukthankar. – 2015 – Режим доступу: https://arxiv.org/abs/1511.06085
14. Full resolution image compression with recurrent neural networks [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – G. Toderici, D. Vincent, N. Johnston, S. J. Hwang, D. Minnen, J. Shor, M. Covell. – 2016 – Режим доступу: https://arxiv.org/abs/1608.05148
15. JPEG Image Compression Using CNN [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Suman Kunwar - 2018 – Режим доступу:

https://www.researchgate.net/publication/322239207_JPEG_Image_Compression_Using_CNN

16. CS231n: Convolutional Neural Networks for Visual Recognition. [Електронний ресурс]: [Веб-сайт]. – Електронні дані. Режим доступу: https://cs231n.github.io/convolutional-networks/
17. CS With James. [Електронний ресурс]: [Веб-сайт]. – Електронні дані. Режим доступу: http://cswithjames.com/
18. Image Compression with Neural Networks [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Nick Johnston, David Minnen – 29.09.2016 – Режим доступу: https://research.googleblog.com/2016/09/image-compression-with-neural-networks.html
19. Cloud AI [Електронний ресурс]: [Веб-сайт]. – Електронні дані. Режим доступу: https://cloud.google.com/products/machine-learning/
20. Amazon Machine Learning [Електронний ресурс]: [Веб-сайт]. – Електронні дані. Режим доступу: https://aws.amazon.com/ru/machine-learning/
21. Azure Machine Learning Studio [Електронний ресурс]: [Веб-сайт]. – Електронні дані. Режим доступу: https://studio.azureml.net/
22. Image Coding Fundamentals [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – 08.05.2017 – Режим доступу: http://videocodecs.blogspot.com/2007/05/image-coding-fundamentals_08.html
23. CRM Software Market Share Report [Електронний ресурс]: [Веб-сайт]. – Електронні дані. Режим доступу: http://www.crmsearch.com/crm-software-market-share.php
24. MS Sharepoint [Електронний ресурс]: [Веб-сайт]. – Електронні дані. Режим доступу: https://products.office.com/en-us/sharepoint/collaboration
25. Олашин О. О. Система покращення алгоритмів стиску зображення // Міжнародний науковий журнал "Інтернаука". — 2018. — №9.

26. Caffe2 [Електронний ресурс]: [Веб-сайт]. – Електронні дані. Режим доступу: https://caffe2.ai/

ДОДАТОК А ЛІСТИНГ КОДУ

HttpHelper.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.IO;

namespace Plugin_SPConnector
{
   internal static class HttpHelper
   {
      /// <summary>
      /// Sends a JSON OData request appending SPO auth cookies to the request header.
      /// </summary>
      /// <param name="uri">The request uri</param>
      /// <param name="method">The http method</param>
      /// <param name="requestContent">A stream containing the request content</param>
      /// <param name="clientHandler">The request client handler</param>
      /// <param name="authUtility">An instance of the auth helper to perform authenticated calls to SPO</param>
      /// <param name="headers">The http headers to append to the request</param>
      public static byte[] SendODataJsonRequest(Uri uri, String method, byte[] requestContent, HttpWebRequest clientHandler, SpoAuthUtility authUtility, Dictionary<string, string> headers = null)
      {

         if (authUtility.UseIntegratedSecurity)
         {
            // do we have a custom user set?
            // - if not then we use the default credentials
            // - in case we have, then we get a new credentials object based on the custom user
            if (!authUtility.IsUserSet)
               clientHandler.UseDefaultCredentials = true;
            else
               clientHandler.Credentials = authUtility.WindowsCredentials;
         }
         else if (clientHandler.CookieContainer == null)
         {
            clientHandler.CookieContainer = new CookieContainer();

            CookieContainer cookieContainer = authUtility.GetCookieContainer(); // get the auth cookies from SPO after authenticating with Microsoft Online Services STS

            foreach (Cookie c in cookieContainer.GetCookies(uri))
            {
               clientHandler.CookieContainer.Add(uri, c); // apppend SPO auth cookies to the request
            }
         }



         return SendHttpRequest(
            uri,
            method,
            requestContent,
            "application/json;odata=verbose;charset=utf-8", // the http content type for the JSON flavor of SP REST services
```

```
                clientHandler,
                headers);
        }


        /// <summary>
        /// Sends an http request to the specified uri and returns the response as a byte array
        /// </summary>
        /// <param name="uri">The request uri</param>
        /// <param name="method">The http method</param>
        /// <param name="requestContent">A stream containing the request content</param>
        /// <param name="contentType">The content type of the http request</param>
        /// <param name="clientHandler">The request client handler</param>
        /// <param name="headers">The http headers to append to the request</param>
        public static byte[] SendHttpRequest(Uri uri, String method, byte[] requestContent = null, string
contentType = null, HttpWebRequest clientHandler = null, Dictionary<string, string> headers = null)
        {
            HttpWebRequest request = clientHandler == null ? (HttpWebRequest)HttpWebRequest.Create(uri) :
clientHandler;

            byte[] responseStream = null;

            request.Method = method;
            request.Accept = contentType;
            request.UserAgent = "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0)";
// This must be here as you will receive 403 otherwise
            request.AllowAutoRedirect = false; // This is key, otherwise it will redirect to failed login SP page

            // append additional headers to the request
            if (headers != null)
            {
                foreach (var header in headers)
                {
                    if (request.Headers.AllKeys.Contains(header.Key))
                    {
                        request.Headers.Remove(header.Key);
                    }

                    request.Headers.Add(header.Key, header.Value);
                }
            }


            if (requestContent != null && (method == "POST" || method == "PUT" || method == "DELETE"))
            {
                if (!string.IsNullOrEmpty(contentType))
                {
                    request.ContentType = contentType; // if the request has a body set the MIME type
                }

                request.ContentLength = requestContent.Length;
                using (Stream s = request.GetRequestStream())
                {
                    s.Write(requestContent, 0, requestContent.Length);
                    s.Close();

                }
            }
            // for all other operations, force the contentlength to be 0. Otherwise you might get an 411 error
            else request.ContentLength = 0;

            // Get the response stream and turn it into a byte[]
```

```
            HttpWebResponse response = (HttpWebResponse)request.GetResponse();
            using (MemoryStream memoryStream = new MemoryStream())
            {
               CopyStream(response.GetResponseStream(), memoryStream);
               if (memoryStream.Length > 0)
               {
                  responseStream = memoryStream.ToArray();
               }
            }

            return responseStream;
         }


         /// <summary>
         /// Copy one stream to the other (this is done in small 32KB chunks)
         /// </summary>
         /// <param name="source">incoming stream</param>
         /// <param name="destination">outgoing stream</param>
         private static void CopyStream(Stream source, Stream destination)
         {
            byte[] buffer = new byte[32768];
            int bytesRead;
            do
            {
               bytesRead = source.Read(buffer, 0, buffer.Length);
               destination.Write(buffer, 0, bytesRead);
            } while (bytesRead != 0);
         }
      }
   }
```

Plugin_Main.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Activities;
using System.ServiceModel;
using System.IO;
using System.Web;
using System.Net;
using System.Security;
using System.Reflection;

using Microsoft.Xrm.Sdk;
using Microsoft.Xrm.Sdk.Messages;
using Microsoft.Xrm.Sdk.Query;
using Microsoft.Xrm.Sdk.Workflow;
using Microsoft.Xrm.Sdk.Client;




namespace Plugin_SPConnector
{
   public class Plugin_Main : IPlugin
   {
```

```
        public IPluginExecutionContext _context;
        public IOrganizationServiceFactory _serviceFactory;
        public IOrganizationService _service;
        public ITracingService _tracingService;

        public void Execute(IServiceProvider serviceProvider)
        {
            _tracingService = (ITracingService)serviceProvider.GetService(typeof(ITracingService));
            _context = (IPluginExecutionContext)
                serviceProvider.GetService(typeof(IPluginExecutionContext));
            _serviceFactory =
(IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrganizationServiceFactory));
            _service = _serviceFactory.CreateOrganizationService(_context.UserId);
            Entity target = new Entity();
            EntityReference targetReference = new EntityReference();
            if (_context.InputParameters.Contains("Target") && _context.InputParameters["Target"] is Entity)
            {
                target = (Entity)_context.InputParameters["Target"];
            }
            if (_context.InputParameters.Contains("Target") && _context.InputParameters["Target"] is
EntityReference)
            {
                targetReference = (EntityReference)_context.InputParameters["Target"];
            }
            _tracingService.Trace("- - - - - - S T A R T - - - - - -");
            string txt = targetReference != null ? "Not Null" : null;
            _tracingService.Trace("targetReference " + txt);
            //_tracingService.Trace(_service != null ? "Not Null" : null);
            switch (_context.MessageName.ToUpper())
            {
                //case "RETRIEVE":
                //    _tracingService.Trace("- - - - - - RETRIVE - - - - - -"); Retrieve(target, _context);
                //    break;
                //case "RETRIEVEMULTIPLE":
                //    if (_context.Stage == 40) RetrieveMultiple(targetColl, _context);
                //    break;
                case "CREATE":
                    _tracingService.Trace("- - - - - - CREATE - - - - - -"); Create(target, _service, _tracingService);
                    break;
                case "DELETE":
                    _tracingService.Trace("- - - - - - DELETE - - - - - -"); Delete(targetReference, _context);
                    break;
            }

            _tracingService.Trace("- - - - - - E N D - - - - - -");
        }
        private void Retrieve(Entity target, IPluginExecutionContext context)
        {
            ModifyRecordForRetrieval(target, true);
        }

        private void RetrieveMultiple(EntityCollection resultCollection, IPluginExecutionContext context)
        {
            if (resultCollection != null)
            {
                foreach (Entity result in resultCollection.Entities)
                    ModifyRecordForRetrieval(result, false);
            }
        }

        private void ModifyRecordForRetrieval(Entity target, bool retrieveDocumentBody)
        {
```

```
            bool isDocument = false;
            string fileName = null;
            string documentBody = null;

            isDocument = target.Contains("isdocument") ? (target.GetAttributeValue<string>("isdocument") ==
"1") : false;
            fileName = target.Contains("filename") ? target.GetAttributeValue<string>("filename") : null;
            documentBody = target.Contains("documentbody") ?
target.GetAttributeValue<string>("documentbody") : null;

            if (isDocument)
            {
               if (!string.IsNullOrEmpty(fileName) && !string.IsNullOrEmpty(documentBody))
               {
                  // make a friendly filename
                  fileName = fileName.Trim();
                  if (fileName.EndsWith(".url", StringComparison.InvariantCultureIgnoreCase))
                  {
                     target.Attributes["filename"] = fileName.Remove(fileName.Length - 4);
                  }
               }

               if (retrieveDocumentBody)
               {

                  // using a different constructot gives you access to sharepoint online
                  SharePointConnector connector = new
SharePointConnector("https://tcomtechtest.sharepoint.com/", "aolashyn@tcomtechtest.onmicrosoft.com",
"Xkda820avt1");

                  // strip the sharepoint url reference from the .url file in the documentBody
                  string[] tokens =
UnicodeEncoding.Unicode.GetString(Convert.FromBase64String(documentBody)).Split('\n');

                  foreach (string token in tokens)
                  {
                     if (token.Trim().StartsWith("URL=", StringComparison.InvariantCultureIgnoreCase))
                     {
                        documentBody = token.Trim().Remove(0, 4);
                        break;
                     }
                  }

                  // get the file from SharePoint
                  byte[] fileBytes = connector.GetFile(documentBody);
                  target.Attributes["documentbody"] = Convert.ToBase64String(fileBytes);
               }
            }
         }

         private void Create(Entity target, IOrganizationService _service, ITracingService _tracingService)
         {

            EntityReference objectRef = (EntityReference)target["objectid"];
            string entityName = objectRef.LogicalName;
            _tracingService.Trace("entityName " + entityName);

            // != -> == !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
            if ((entityName != "new_inspection_certificate")||(entityName != "new_car"))
            {

               bool isDocument = false;
```

```
string fileName = "";
string documentBody = "";

isDocument = (bool)target["isdocument"] ? (bool)target["isdocument"] : false;
_tracingService.Trace("isDocument " + isDocument.ToString());
fileName = (bool)target.Contains("filename") ? (String)target["filename"] : null;
_tracingService.Trace("fileName " + fileName.ToString());
documentBody = (bool)target.Contains("documentbody") ? (String)target["documentbody"] : null;
_tracingService.Trace("documentBody " + documentBody.ToString());

if (isDocument)
{
   if (!string.IsNullOrEmpty(fileName) && !string.IsNullOrEmpty(documentBody))
   {
      fileName = fileName.Trim();
      byte[] fileBytes = Convert.FromBase64String(documentBody);

      Entity app = new Entity();
      app = GetSPData();
      _tracingService.Trace("after response GetSPdata " + app["new_login"]);

      //getting folder location

      RetrieveEntityRequest request2 = new RetrieveEntityRequest();
      request2.LogicalName = entityName;
      RetrieveEntityResponse response2 = (RetrieveEntityResponse)_service.Execute(request2);
      string mainField = response2.EntityMetadata.PrimaryNameAttribute;
      int objecttypecode = response2.EntityMetadata.ObjectTypeCode.Value;


      ColumnSet col = new ColumnSet(mainField, entityName + "id");
      Entity ent = _service.Retrieve(entityName, objectRef.Id, col);


      string id = ent.Id.ToString().Replace("-", "").ToUpper();
      string folderName = DeleteCharacters(ent[mainField].ToString());

      string folder = "/" + entityName + "/" + folderName + "_" + id + "/";
      _tracingService.Trace("folder " + folder);
      string folderShort = ent[mainField] + "_" + ent.Id.ToString();

      _tracingService.Trace(folder);

      // add the file to SharePoint, return the unique name

      SharePointConnector connector = new
SharePointConnector((String)app["new_sharepointurl"], (String)app["new_login"], (String)app["new_password"]);

      // build the dictionary to pass the values to be stored inside the list item
      Dictionary<string, object> dict = new Dictionary<string, object>();
      dict.Add("MoreText", "Some more text");
      dict.Add("Description0", "A very long description spanning multiple lines");
      dict.Add("SomeNumber", 13);
      dict.Add("MyChoice", "Keuze 3");

      _tracingService.Trace("-----------AddFile-----------------");
      string sharepointfileName = connector.AddFileAndFolder(fileName,
(String)app["new_sharepointurl"] + folder, folder, fileBytes, dict, _tracingService);
      _tracingService.Trace("-----------End AddFile-----------------");

      documentBody = string.Format(sharepointfileName);
      _tracingService.Trace("Doc " + documentBody);
```

```
                    target["notetext"] = documentBody;
                    target.EntityState = null;
                    _tracingService.Trace("file delete ");
                    target["documentbody"] = null;
                    target["filename"] = null;
                    target["filesize"] = null;
                    target["isdocument"] = false;
                    //target["documentbody"] = "";
                    _service.Update(target);
                }
            }
        }
    }
    private void Delete(EntityReference targetreference, IPluginExecutionContext context)
    {
        Entity entity = (Entity)_context.PreEntityImages["Target"];
        _tracingService.Trace(entity.ToString());
        Entity app = new Entity();
        app = GetSPData();
        if (entity != null)
        {
            string notetext = entity.Contains("notetext") ? entity.GetAttributeValue<string>("notetext") : null;

            if (!string.IsNullOrEmpty(notetext))
            {

                SharePointConnector connector = new SharePointConnector((String)app["new_sharepointurl"],
(String)app["new_login"], (String)app["new_password"]);
                connector.DeleteFile(notetext);
            }
        }
    }

    private Entity GetSPData()
    {
        QueryExpression query = new QueryExpression();
        _tracingService.Trace("query");
        query.EntityName = "new_spsetting";
        _tracingService.Trace("ColumnSet");
        query.ColumnSet = new ColumnSet("new_login", "new_password", "new_sharepointurl");
        EntityCollection entitycoll = new EntityCollection();              //  Коллекция сущностей
        RetrieveMultipleResponse response = new RetrieveMultipleResponse(); //  Ответ
        RetrieveMultipleRequest request = new RetrieveMultipleRequest();    //  Запрос
        request = new RetrieveMultipleRequest();
        request.Query = query;
        response = (RetrieveMultipleResponse)_service.Execute(request);
        _tracingService.Trace("GetSPdata " + response.EntityCollection.Entities[0]["new_login"]);
        return response.EntityCollection.Entities[0];
    }

    private string DeleteCharacters(string str)
    {
        string res = str;
        res = str.Replace("\"", "");
        res = res.Replace("\'", "");
        res = res.Replace("#", "");
        res = res.Replace("&", "");
        res = res.Replace("?", "");
        res = res.Replace(":", "");
        res = res.Replace(";", "");
        return res;
```

```
        }
    }
}
```

SharePointConnector.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Xml.Linq;
using System.Xml.XPath;
using System.Runtime.Serialization.Json;
using System.Web.Script.Serialization;
using Microsoft.Xrm.Sdk;

namespace Plugin_SPConnector
{
    public class SharePointConnector
    {
        private bool _hasCookie = false;

        //      /// <summary>
        //      /// Use SharePoint 2013 on-premise environment
        //      /// </summary>
        //      /// <param name="siteUrl"></param>
        //      public SharePointConnector(string siteUrl)
        //      {
        //          if (siteUrl.EndsWith("/")) siteUrl = siteUrl.Substring(0, siteUrl.Length - 1);
        //          Uri spSite = new Uri(siteUrl);
        //          _hasCookie = SpoAuthUtility.Create(spSite, "", "", true);

        //      }

        //      /// <summary>
        //      /// Use SharePoint onpremise or online environment
        ///// in case of using on premise, the passed in credentials will be used to impersonate the user call
        //      /// </summary>
        //      /// <param name="siteUrl">url of the site</param>
        //      /// <param name="userName">domain name of the user</param>
        //      /// <param name="password">password of the user</param>
        //      /// <param name="isOnline">use sharepoint online?</param>
        //      public SharePointConnector(string siteUrl, string userName, string password, bool isOnline)
        //      {
        //          if (siteUrl.EndsWith("/")) siteUrl = siteUrl.Substring(0, siteUrl.Length - 1);
        //          Uri spSite = new Uri(siteUrl);

        //              _hasCookie = (!isOnline) ? SpoAuthUtility.Create(spSite, userName, password, true) : SpoAuthUtility.Create(spSite, userName, WebUtility.HtmlEncode(password), false); ;
        //      }



        /// <summary>
        /// Use SharePoint online environment
        /// </summary>
        /// <param name="siteUrl"></param>
```

```
        /// <param name="userEmailAddress"></param>
        /// <param name="password"></param>
        public SharePointConnector(string siteUrl, string userEmailAddress, string password)
        {
            if (siteUrl.EndsWith("/")) siteUrl = siteUrl.Substring(0, siteUrl.Length - 1);
            Uri spSite = new Uri(siteUrl);
            _hasCookie = SpoAuthUtility.Create(spSite, userEmailAddress, WebUtility.HtmlEncode(password),
false);
        }

        /// <summary>
        /// Break the item level permission inheritance.
        /// </summary>
        /// <param name="filefolderName">name of file</param>
        /// <param name="doclib">documentlibrary</param>
        private void BreakItemLevelPermissionInheritance(string filefolderName, string doclib, bool isFile =
true)
        {
            var formDigest = GetFormDigest();

            // break inheritance
            string restQuery = "";
            if (isFile)
                restQuery                                                                                    =
string.Format("_api/web/getfilebyserverrelativeurl('{0}/{1}')/ListItemAllFields/breakroleinheritance(false)",
GetRelativeUrl(doclib), filefolderName);
            else
                restQuery                                                                                    =
string.Format("_api/web/getfolderbyserverrelativeurl('{0}/{1}')/ListItemAllFields/breakroleinheritance(false)",
GetRelativeUrl(doclib), filefolderName);


            Uri url = new Uri(String.Format("{0}/{1}", SpoAuthUtility.Current.SiteUrl, restQuery));
            Dictionary<string, string> headers = new Dictionary<string, string>();
            headers.Add("X-RequestDigest", formDigest);

            // Send a json odata request to SPO rest services to fetch all list items for the list.
            byte[] result = HttpHelper.SendODataJsonRequest(
            url,
            "POST",                                 // reading data from SP through the rest api usually uses the GET
verb
            null,
            (HttpWebRequest)HttpWebRequest.Create(url),
            SpoAuthUtility.Current,                 // pass in the helper object that allows us to make authenticated
calls to SPO rest services
            headers                                 // specify that sharepoint needs to return uncontaminated bytes....
            );

            // not used for debugging purposes
            var resultstring = System.Text.UTF8Encoding.UTF8.GetString(result);
        }


        /// <summary>
        /// clear all permissions from item
        /// </summary>
        /// <param name="fileFolderName"></param>
        /// <param name="doclib"></param>
        private void ClearItemLevelPermissions(string fileFolderName, string doclib, bool isFile = true)
        {
            var formDigest = GetFormDigest();
```

```
                // break inheritance
                string restQuery = "";
                if (isFile)
                    restQuery                                                                                    =
string.Format("_api/web/getfilebyserverrelativeurl('{0}/{1}')/ListItemAllFields/roleassignments",
GetRelativeUrl(doclib), fileFolderName);
                else
                    restQuery                                                                                    =
string.Format("_api/web/getfolderbyserverrelativeurl('{0}/{1}')/ListItemAllFields/roleassignments",
GetRelativeUrl(doclib), fileFolderName);

                Uri url = new Uri(String.Format("{0}/{1}", SpoAuthUtility.Current.SiteUrl, restQuery));
                Dictionary<string, string> headers = new Dictionary<string, string>();
                headers.Add("X-RequestDigest", formDigest);
                headers.Add("X-HTTP-Method", "DELETE");

                // Send a json odata request to SPO rest services to fetch all list items for the list.
                byte[] result = HttpHelper.SendODataJsonRequest(
                url,
                "POST",                             // reading data from SP through the rest api usually uses the GET
verb
                null,
                (HttpWebRequest)HttpWebRequest.Create(url),
                SpoAuthUtility.Current,                 // pass in the helper object that allows us to make authenticated
calls to SPO rest services
                headers                             // specify that sharepoint needs to return uncontaminated bytes....
                );

                // not used for debugging purposes
                var resultstring = System.Text.UTF8Encoding.UTF8.GetString(result);

            }

            /// <summary>
            /// Get the sharepoint id for a given sharepoint role
            /// </summary>
            /// <param name="roleName">name of the sharepoint role</param>
            /// <returns>id</returns>
            private int GetRoleDefinitionId(string roleName = "Contribute")
            {
                string restQuery = string.Format("_api/web/roledefinitions/getbyname('{0}')/Id", roleName);

                Uri url = new Uri(String.Format("{0}/{1}", SpoAuthUtility.Current.SiteUrl, restQuery));
                Dictionary<string, string> headers = new Dictionary<string, string>();

                // Send a json odata request to SPO rest services to fetch all list items for the list.
                byte[] result = HttpHelper.SendODataJsonRequest(
                url,
                "GET",                              // reading data from SP through the rest api usually uses the GET
verb
                null,
                (HttpWebRequest)HttpWebRequest.Create(url),
                SpoAuthUtility.Current,                 // pass in the helper object that allows us to make authenticated
calls to SPO rest services
                headers                             // specify that sharepoint needs to return uncontaminated bytes....
                );

                // get the result and strip out all tags
                var resultstring = GetValueFromJSON(System.Text.UTF8Encoding.UTF8.GetString(result), "d",
"Id");

                return int.Parse(resultstring);
```

```
        }


        /// <summary>
        /// Get the value from json blob
        /// </summary>
        /// <param name="json"></param>
        /// <param name="rootKey"></param>
        /// <param name="key"></param>
        /// <returns></returns>
        private string GetValueFromJSON(string json, string rootKey, string key)
        {
            string result = null;

            var deserializer = new JavaScriptSerializer();

            var wrapper = (Dictionary<string, object>)deserializer.DeserializeObject(json);
            if (wrapper.ContainsKey(rootKey))
            {
                var collection = (Dictionary<string, object>)wrapper[rootKey];
                if (collection.ContainsKey(key))
                {
                    result = collection[key].ToString();
                }
            }
            return result;
        }

        ///// <summary>
        /// Get the userId by using the loginname
        ///
        /// SharePoint environment
        /// ------------------------
        /// Example login name format
        /// SharePoint Online, or SharePoint 2013 on-premises using forms --> i:0#.f|membership|user@domain.com --> …/users(@v)?@v='i%3A0%23.f%7Cmembership%7Cuser%40domain.onmicrosoft.com'
        /// SharePoint 2013 on-premises using Windows claims --> i:0#.w|domain\user --> …/users(@v)?@v='i%3A0%23.w%7Cdomain\user'
        /// SharePoint 2013 on-premises using SAML claims --> i:05:t|adfs with roles|user@domain.com --> …/users(@v)?@v='i%3A05%3At%7Cadfs+with+roles%7Cuser%40domain.com'
        /// </summary>
        /// <param name="userName"></param>
        /// <returns></returns>
        private int GetUserId(string userName)
        {
            // use windows claims
            string restQuery = string.Format("_api/web/siteusers(@v)/Id?@v='i%3A0%23.w%7C{0}'", userName);

            Uri url = new Uri(String.Format("{0}/{1}", SpoAuthUtility.Current.SiteUrl, restQuery));

            Dictionary<string, string> headers = new Dictionary<string, string>();

            // Send a json odata request to SPO rest services to fetch all list items for the list.
            byte[] result = HttpHelper.SendODataJsonRequest(
            url,
            "GET",                          // reading data from SP through the rest api usually uses the GET verb
            null,
            (HttpWebRequest)HttpWebRequest.Create(url),
```

```
                SpoAuthUtility.Current,                    // pass in the helper object that allows us to make authenticated
calls to SPO rest services
                headers                                // specify that sharepoint needs to return uncontaminated bytes....
                );

                // get the result and strip out all tags
                var resultstring = GetValueFromJSON(System.Text.UTF8Encoding.UTF8.GetString(result), "d",
"Id");

                return int.Parse(resultstring);

            }


            /// <summary>
            /// Set the item level permissions for the user
            /// </summary>
            /// <param name="fileName">relative name of file or folder</param>
            /// <param name="doclib">absolute url of documentlibrary</param>
            /// <param name="user">loginname of user</param>
            /// <param name="roleId">sharepoint id of the role</param>
            /// <param name="isFile">if true then it is a file. if false it is as folder</param>
            private void SetItemLevelPermissionsForUser(string fileName, string doclib, string user, int roleId, bool
isFile)
            {
                try
                {

                    var userId = GetUserId(user);

                    var formDigest = GetFormDigest();

                    // break inheritance
                    string restQuery = "";
                    if (isFile)
                        restQuery                                                                          =
string.Format("_api/web/getfilebyserverrelativeurl('{0}/{1}')/ListItemAllFields/roleassignments/addroleassignment(
principalid={2}, roleDefId={3})", GetRelativeUrl(doclib), fileName, userId, roleId);
                    else
                        restQuery                                                                          =
string.Format("_api/web/getfolderbyserverrelativeurl('{0}/{1}')/ListItemAllFields/roleassignments/addroleassignme
nt(principalid={2}, roleDefId={3})", GetRelativeUrl(doclib), fileName, userId, roleId);


                    Uri url = new Uri(String.Format("{0}/{1}", SpoAuthUtility.Current.SiteUrl, restQuery));
                    Dictionary<string, string> headers = new Dictionary<string, string>();
                    headers.Add("X-RequestDigest", formDigest);


                    // Send a json odata request to SPO rest services to fetch all list items for the list.
                    byte[] result = HttpHelper.SendODataJsonRequest(
                    url,
                    "POST",                                // reading data from SP through the rest api usually uses the GET
verb
                    null,
                    (HttpWebRequest)HttpWebRequest.Create(url),
                    SpoAuthUtility.Current,                    // pass in the helper object that allows us to make authenticated
calls to SPO rest services
                    headers                                // specify that sharepoint needs to return uncontaminated bytes....
                    );

                    // not used for debugging purposes
```

```
            var resultstring = System.Text.UTF8Encoding.UTF8.GetString(result);
        }
        catch (Exception e)
        {
            // ignore.. the user doesnt have access rights on sharepoint (or doesn't exist there)
        }
    }


    /// <summary>
    ///
    /// </summary>
    /// <param name="fileName">relative name of file or folder</param>
    /// <param name="doclib">absolute url of documentlibrary</param>
    /// <param name="user">loginname of user</param>
    /// <param name="isFile">if true then it is a file. if false it is as folder</param>
    /// <returns></returns>
    public bool SetItemLevelPermissions(string fileName, string doclib, string[] users, bool isFile = true)
    {
        try
        {
            // break role inheritance
            BreakItemLevelPermissionInheritance(fileName, doclib, isFile);

            // clear any existing permission from the file
            //  ClearItemLevelPermissions(fileName, doclib);

            // construct a new list of permissions and assign it to the list of users
            int contributeRoleId = GetRoleDefinitionId();

            // for each user in the list, set the role id
            foreach (var user in users)
            {
                SetItemLevelPermissionsForUser(fileName, doclib, user, contributeRoleId, isFile);
            }


            return true;
        }
        catch (Exception e)
        {
            throw new Exception(string.Format("SharePoint item level permissions could not be set on
document library: {0} for folder {1}.", doclib, fileName));

        }


    }

    /// <summary>
    /// Get a unique file name for a file within a document library
    /// </summary>
    /// <param name="fileName">name of the fle</param>
    /// <param name="doclib">absolute url of documentlibrary</param>
    /// <returns>unique file name</returns>
    public string GetUniqueFileName(string fileName, string doclib)
    {
        try
        {
            if (_hasCookie)
            {
```

```
                    string restQuery = string.Format("_api/web/getfilebyserverrelativeurl('{0}/{1}')/Exists",
GetRelativeUrl(doclib), fileName);

                    Uri url = new Uri(String.Format("{0}/{1}", SpoAuthUtility.Current.SiteUrl, restQuery));
                    Dictionary<string, string> headers = new Dictionary<string, string>();

                    // Send a json odata request to SPO rest services to fetch all list items for the list.
                    byte[] result = HttpHelper.SendODataJsonRequest(
                    url,
                    "GET",                          // reading data from SP through the rest api usually uses the
GET verb
                    null,
                    (HttpWebRequest)HttpWebRequest.Create(url),
                    SpoAuthUtility.Current,                 // pass in the helper object that allows us to make
authenticated calls to SPO rest services
                    headers                     // specify that sharepoint needs to return uncontaminated bytes....
                    );

                    string resultstring = System.Text.UTF8Encoding.UTF8.GetString(result);

                    // if the file already exists, insert a timestamp
                    if (resultstring.IndexOf("true") > -1)
                    {
                       int pos = fileName.LastIndexOf('.');
                       if (pos > -1) fileName = fileName.Insert(pos, string.Format(" ({0:dd-MM-yyyy HHmmss})",
DateTime.Now));

                    }
                 }

              }
              catch
              {
                 // do nothing. appearantly the file does not exist
              }
              return fileName;
           }

           /// <summary>
           /// Add a file to a document library
           /// </summary>
           /// <param name="fileName">name of file</param>
           /// <param name="doclib">absolute url of documentlibrary</param>
           /// <param name="fileData">bytes containing file</param>B
           /// <param name="fieldData">meta data for file</param>
           /// <returns></returns>
           public string AddFile(string fileName, string doclib, byte[] fileData, Dictionary<string, object>
fieldData, ITracingService _tracingService) //delete tracing service
           {
              Guid newID = Guid.Empty;

              string formDigest = GetFormDigest();

              if (_hasCookie && !string.IsNullOrEmpty(formDigest))
              {
                 fileName = GetUniqueFileName(fileName, doclib);
                 string                                    restQuery                                    =
string.Format("_api/web/getfolderbyserverrelativeurl('{0}')/files/add(overwrite=true,                 url='{1}')",
GetRelativeUrl(doclib), fileName);

                 Uri url = new Uri(String.Format("{0}/{1}", SpoAuthUtility.Current.SiteUrl, restQuery));
                 Dictionary<string, string> headers = new Dictionary<string, string>();
```

```
                    headers.Add("X-RequestDigest", formDigest);

                    // Send a json odata request to SPO rest services to fetch all list items for the list.
                    byte[] result = HttpHelper.SendODataJsonRequest(
                    url,
                    "POST",                             // reading data from SP through the rest api usually uses the GET
verb
                    fileData,
                    (HttpWebRequest)HttpWebRequest.Create(url),
                    SpoAuthUtility.Current,             // pass in the helper object that allows us to make authenticated
calls to SPO rest services
                    headers                             // specify that sharepoint needs to return uncontaminated bytes....
                    );

                    string resultstring = System.Text.UTF8Encoding.UTF8.GetString(result);
                    //_tracingService.Trace("resulting string " + resultstring);
                    // get the full path to be returned
                    string resturl = "";
                    // Get the returning REST query, to get all fields
                    // get also the type of the sharepoint item, otherwise we cannot set the data
                    int pos = resultstring.IndexOf("\"ServerRelativeUrl\":\"");
                    if (pos > -1)
                    {
                       pos += 21;
                       int endpos = resultstring.IndexOf("\",", pos);
                       if (endpos > pos)
                       {
                          resturl = string.Format("{0}/{1}", SpoAuthUtility.Current.SiteUrl, resultstring.Substring(pos
+ 1, endpos - pos - 1));
                       }
                    }



                    // Get the returning REST query, to get all fields
                    pos = resultstring.IndexOf("\"ListItemAllFields\":{\"__deferred\":{\"uri\":\"");
                    if (pos > -1)
                    {
                       pos += 42;
                       int endpos = resultstring.IndexOf("\"}},", pos);
                       if (endpos > pos)
                          resultstring = resultstring.Substring(pos, endpos - pos);
                       else
                          resultstring = "";

                    }
                    else { resultstring = ""; }

                    _tracingService.Trace("Rest url " + resturl.ToString());
                    // update the list item
                    //UpdateListItem(resultstring, fieldData, formDigest,_tracingService);

                    _tracingService.Trace("Rest url " + resturl.ToString());
                    // return the new url
                    return resturl;
                 }

                // return nothing...
                return string.Empty;

             }
```

```
        private bool FolderExists(string doclib, string formDigest) // checking existance of folder
        {
            bool res = false;
            string folderExistQuery = string.Format("_api/web/getfolderbyserverrelativeurl('{0}')/Exists",
GetRelativeUrl(doclib));
            Uri url1 = new Uri(String.Format("{0}/{1}", SpoAuthUtility.Current.SiteUrl, folderExistQuery));
            Dictionary<string, string> headers1 = new Dictionary<string, string>();
            headers1.Add("X-RequestDigest", formDigest);
            byte[] result1 = HttpHelper.SendODataJsonRequest(
                url1,
                "GET",
                null,
                (HttpWebRequest)HttpWebRequest.Create(url1),
                SpoAuthUtility.Current,
                headers1
                );

            string resu = System.Text.UTF8Encoding.UTF8.GetString(result1).Remove(1, 14);
            resu = resu.Replace("}", "");
            resu = resu.Replace("{", "");
            res = Convert.ToBoolean(resu);
            return res;
        }

        //returned ServerRelativeUrl of all files in folder
        public List<string> GetFilesFromFolder(string doclib, string formDigest, ITracingService
_tracingService)
        {
            List<string> list = new List<string>();
            string folderItemsQuery = string.Format("_api/web/getfolderbyserverrelativeurl('{0}')/Files",
GetRelativeUrl(doclib));
            Uri url1 = new Uri(String.Format("{0}/{1}", SpoAuthUtility.Current.SiteUrl, folderItemsQuery));
            Dictionary<string, string> headers1 = new Dictionary<string, string>();
            headers1.Add("X-RequestDigest", formDigest);
            byte[] result1 = HttpHelper.SendODataJsonRequest(
                url1,
                "GET",
                null,
                (HttpWebRequest)HttpWebRequest.Create(url1),
                SpoAuthUtility.Current,
                headers1
                );
            var jsonReader = JsonReaderWriterFactory.CreateJsonReader(result1, new
System.Xml.XmlDictionaryReaderQuotas());
            var root = XElement.Load(jsonReader);

            IEnumerable<XElement> list1 = root.XPathSelectElements("//ServerRelativeUrl");
            int i = 0;
            foreach (XElement el in list1)
            {
                _tracingService.Trace("el " + el.Value);
                list.Add(el.Value);
                _tracingService.Trace("list " + list[i]);
                i++;
            }

            return list;
        }

        private void CreateFolder(string doclib,string folder, string formDigest) // creating folder
        {
```

```
            string folderCreateQuery = string.Format("_api/web/getfolderbyserverrelativeurl('Shared Documents')/folders/add('{0}')", GetRelativeUrl(doclib));
            Uri url2 = new Uri(String.Format("{0}/{1}", SpoAuthUtility.Current.SiteUrl, folderCreateQuery));
            Dictionary<string, string> headers2 = new Dictionary<string, string>();
            headers2.Add("X-RequestDigest", formDigest);
            byte[] result2 = HttpHelper.SendODataJsonRequest(
               url2,
               "POST",
               null,
               (HttpWebRequest)HttpWebRequest.Create(url2),
               SpoAuthUtility.Current,
               headers2
               );
         }

         public string AddFileAndFolder(string fileName, string doclib,string folder, byte[] fileData, Dictionary<string, object> fieldData, ITracingService _tracingService) //delete tracing service
         {
            Guid newID = Guid.Empty;

            string formDigest = GetFormDigest();

            if (_hasCookie && !string.IsNullOrEmpty(formDigest))
            {
               bool isExists = FolderExists(doclib, formDigest);
               _tracingService.Trace("result "+ isExists);
               if (!isExists)
               {
                  CreateFolder(doclib,folder, formDigest);
                  _tracingService.Trace("Create folder");
               }

               //throw new InvalidPluginExecutionException("After creating "+FolderExists(doclib,formDigest));
               fileName = GetUniqueFileName(fileName, doclib);
               string restQuery = string.Format("_api/web/getfolderbyserverrelativeurl('{0}')/files/add(overwrite=true, url='{1}')", GetRelativeUrl(doclib), fileName);

               Uri url = new Uri(String.Format("{0}/{1}", SpoAuthUtility.Current.SiteUrl, restQuery));
               Dictionary<string, string> headers = new Dictionary<string, string>();
               headers.Add("X-RequestDigest", formDigest);

               // Send a json odata request to SPO rest services to fetch all list items for the list.
               byte[] result = HttpHelper.SendODataJsonRequest(
               url,
               "POST",                     // reading data from SP through the rest api usually uses the GET verb
               fileData,
               (HttpWebRequest)HttpWebRequest.Create(url),
               SpoAuthUtility.Current,     // pass in the helper object that allows us to make authenticated calls to SPO rest services
               headers                     // specify that sharepoint needs to return uncontaminated bytes....
               );
               var jsonReader = JsonReaderWriterFactory.CreateJsonReader(result, new System.Xml.XmlDictionaryReaderQuotas());
               var root = XElement.Load(jsonReader);

               string resultstring = root.XPathSelectElement("//ServerRelativeUrl").Value;
               return resultstring;
               //throw new InvalidPluginExecutionException("resulting string " + resultstring);
               ////_tracingService.Trace("resulting string " + resultstring);
```

```
                    //// get the full path to be returned
                    //string resturl = "";
                    //// Get the returning REST query, to get all fields
                    //// get also the type of the sharepoint item, otherwise we cannot set the data
                    //int pos = resultstring.IndexOf("\"ServerRelativeUrl\":\"");
                    //if (pos > -1)
                    //{
                    //    pos += 21;
                    //    int endpos = resultstring.IndexOf("\",", pos);
                    //    if (endpos > pos)
                    //    {
                    //      resturl = string.Format("{0}/{1}", SpoAuthUtility.Current.SiteUrl, resultstring.Substring(pos
+ 1, endpos - pos - 1));
                    //    }
                    //}



                    //// Get the returning REST query, to get all fields
                    //pos = resultstring.IndexOf("\"ListItemAllFields\":{\"__deferred\":{\"uri\":\"");
                    //if (pos > -1)
                    //{
                    //    pos += 42;
                    //    int endpos = resultstring.IndexOf("\"}},", pos);
                    //    if (endpos > pos)
                    //        resultstring = resultstring.Substring(pos, endpos - pos);
                    //    else
                    //        resultstring = "";

                    //}
                    //else { resultstring = ""; }

                    ////_tracingService.Trace("Rest url " + resturl.ToString());
                    //// update the list item
                    ////UpdateListItem(resultstring, fieldData, formDigest,_tracingService);

                    ////_tracingService.Trace("Rest url " + resturl.ToString());
                    //// return the new url
                    //return resturl;

                }

                // return nothing...
                return string.Empty;

            }

            /// <summary>
            /// Update the list item
            /// </summary>
            /// <param name="result">result query of the retrieval action</param>
            /// <param name="data">dictionary with item fields</param>
            public void UpdateListItem(string result, Dictionary<string, object> data, string formDigest,
ITracingService _tracingService)
            {
                string u = result;
                _tracingService.Trace("--------UpdateListItem----------");
                if (_hasCookie && !string.IsNullOrEmpty(result) && data != null && data.Count > 0)
                {

                    Uri url = new Uri(u);
                    Dictionary<string, string> headers = new Dictionary<string, string>();
```

```
                    // Send a json odata request to SPO rest services to fetch all list items for the list.
                    byte[] output = HttpHelper.SendODataJsonRequest(
                    url,
                    "GET",                              // reading data from SP through the rest api usually uses the GET
verb
                    null,
                    (HttpWebRequest)HttpWebRequest.Create(url),
                    SpoAuthUtility.Current,             // pass in the helper object that allows us to make authenticated
calls to SPO rest services
                    headers                             // specify that sharepoint needs to return uncontaminated bytes....
                    );


                    string resultstring = System.Text.UTF8Encoding.UTF8.GetString(output);
                    //_tracingService.Trace("Resulting string - " + resultstring);
                    string resturl = "";
                    // Get the returning REST query, to get all fields
                    // get also the type of the sharepoint item, otherwise we cannot set the data
                    int pos = resultstring.IndexOf("\"uri\":\"");
                    if (pos > -1)
                    {
                       pos += 7;
                       int endpos = resultstring.IndexOf("\",", pos);
                       if (endpos > pos)
                          resturl = resultstring.Substring(pos, endpos - pos);
                    }

                    string itemtype = "";
                    pos = resultstring.IndexOf("\"type\":\"");
                    if (pos > -1)
                    {
                       pos += 8;
                       int endpos = resultstring.IndexOf("\"}", pos);
                       if (endpos > pos)
                          itemtype = resultstring.Substring(pos, endpos - pos);
                    }



                    // do we have a valid URI for the rest call?
                    // then we can build the json object and send it out
                    if (!string.IsNullOrEmpty(resturl))
                    {

                       // build the json string
                       string stringData = "{'__metadata': {'type':'" + itemtype + "'}, ";
                       bool first = true;
                       foreach (KeyValuePair<string, object> kvp in data)
                       {

                          if (!first) stringData += ",";

                          if (kvp.Value is string)
                          {
                             stringData += string.Format("'{0}':'{1}'", kvp.Key, kvp.Value);
                          }
                          else
                          {
                             stringData += string.Format("'{0}':{1}", kvp.Key, kvp.Value);
                          }
                          first = false;
```

```
                    }

                    stringData += "}";


                    Uri updateUrl = new Uri(resturl.Replace("(guid'", "('"));
                    headers = new Dictionary<string, string>();
                    headers.Add("X-RequestDigest", formDigest);
                    headers.Add("X-HTTP-Method", "MERGE");
                    headers.Add("IF-MATCH", "*");

                    _tracingService.Trace("update url " + updateUrl.ToString());

                    // Send a json odata request to SPO rest services to fetch all list items for the list.
                    byte[] updateresult = HttpHelper.SendODataJsonRequest(
                    updateUrl,
                    "POST",                         // reading data from SP through the rest api usually uses the
GET verb
                    Encoding.UTF8.GetBytes(stringData),          // make a byte array of the rest call
                    (HttpWebRequest)HttpWebRequest.Create(updateUrl),
                    SpoAuthUtility.Current,                     // pass in the helper object that allows us to make
authenticated calls to SPO rest services
                    headers                         // specify that sharepoint needs to return uncontaminated bytes....
                    );
                    _tracingService.Trace("------UpdateItemList END-----------");
                }
            }
        }


        /// <summary>
        /// remove a file from a document library
        /// </summary>
        /// <param name="doclib">absolute url of the file</param>
        /// <returns></returns>
        public bool DeleteFile(string doclib)
        {

            try
            {
                string formDigest = GetFormDigest();

                if (_hasCookie && !string.IsNullOrEmpty(formDigest))
                {
                    string restQuery = string.Format("_api/web/getfilebyserverrelativeurl('{0}')",
GetRelativeUrl(doclib));

                    Uri url = new Uri(String.Format("{0}/{1}", SpoAuthUtility.Current.SiteUrl, restQuery));
                    Dictionary<string, string> headers = new Dictionary<string, string>();
                    headers.Add("X-RequestDigest", formDigest);
                    headers.Add("X-HTTP-Method", "DELETE");

                    // Send a json odata request to SPO rest services to fetch all list items for the list.
                    byte[] result = HttpHelper.SendODataJsonRequest(
                    url,
                    "POST",                         // reading data from SP through the rest api usually uses the
GET verb
                    null,
                    (HttpWebRequest)HttpWebRequest.Create(url),
                    SpoAuthUtility.Current,                     // pass in the helper object that allows us to make
authenticated calls to SPO rest services
                    headers                         // specify that sharepoint needs to return uncontaminated bytes....
```

```
                    );

                    // file deleted
                    return true;
                }
            }
            catch
            {
                // troubles in paradise
                return false;
            }

            // nothing to delete
            return false;
        }


        /// <summary>
        /// Return a byte array containing the file bytes
        /// </summary>
        /// <param name="fileUrl">full url to the file</param>
        /// <returns>byte[] containing file bytes, or null on no results</returns>
        public byte[] GetFile(string fileUrl)
        {

            if (_hasCookie)
            {
                string restQuery = string.Format("_api/web/GetFileByServerRelativeUrl('{0}')/$value",
GetRelativeUrl(fileUrl));

                Uri url = new Uri(String.Format("{0}/{1}", SpoAuthUtility.Current.SiteUrl, restQuery));
                Dictionary<string, string> headers = new Dictionary<string, string>();
                headers.Add("binaryStringResponseBody", "true");

                // Send a json odata request to SPO rest services to fetch all list items for the list.
                byte[] result = HttpHelper.SendODataJsonRequest(
                url,
                "GET",                              // reading data from SP through the rest api usually uses the GET
verb
                null,
                (HttpWebRequest)HttpWebRequest.Create(url),
                SpoAuthUtility.Current,             // pass in the helper object that allows us to make authenticated
calls to SPO rest services
                headers                             // specify that sharepoint needs to return uncontaminated bytes....
                );

                return result;
            }

            return null;
        }



        /// <summary>
        /// Get a relative URL based on the URL passed in e.g.
        /// http://sharepoint:8080/sites/bas/documents/file.docx  results in /sites/bas/documents/file.docx
        /// </summary>
        /// <param name="url">absolute URL (containing http:// )</param>
        /// <returns>relative URL</returns>
        private string GetRelativeUrl(string url)
```

```
        {
            url = new Uri(url).AbsolutePath;
            if (url.EndsWith("/")) url = url.Substring(0, url.Length - 1);
            return url;
        }


        /// <summary>
        /// Get the form digest
        /// required for posting content to sharepoint
        /// </summary>
        /// <returns></returns>
        public string GetFormDigest()
        {
            string newFormDigest = null;

            if (_hasCookie)
            {

                // 1st request to get the context information
                Uri url = new Uri(String.Format("{0}/{1}", SpoAuthUtility.Current.SiteUrl, "_api/contextinfo"));

                byte[] data = HttpHelper.SendODataJsonRequest(
                            url,
                            "POST",
                            null,
                            (HttpWebRequest)HttpWebRequest.Create(url),
                            SpoAuthUtility.Current,                 // pass in the helper object that allows us to make
authenticated calls to SPO rest services
                            null                                    // specify that sharepoint needs to return uncontaminated
bytes....
                            );



                string results = System.Text.UTF8Encoding.UTF8.GetString(data);

                // Get the FormDigest Value
                var startTag = "FormDigestValue";
                var endTag = "LibraryVersion";
                var startTagIndex = results.IndexOf(startTag) + 1;
                var endTagIndex = results.IndexOf(endTag, startTagIndex);

                if ((startTagIndex >= 0) && (endTagIndex > startTagIndex))
                {
                    newFormDigest = results.Substring(startTagIndex + startTag.Length + 2, endTagIndex -
startTagIndex - startTag.Length - 5);
                }
            }
            return newFormDigest;
        }


        /// <summary>
        /// Get the absolute url of the SharePoint document library
        /// </summary>
        /// <param name="fullurl">absolute url passed in</param>
        /// <param name="siteUrl">absolute site url passed in</param>
        /// <returns>absolute url pointing to the document library</returns>
        public string GetDocumentLibraryUrl(string fullurl, string siteUrl)
        {
            if (siteUrl.EndsWith("/")) siteUrl = siteUrl.Substring(0, siteUrl.Length - 1);
```

```
            fullurl = fullurl.ToLowerInvariant().Replace(siteUrl.ToLowerInvariant(), "").Trim();
            if (fullurl.StartsWith("/")) fullurl = fullurl.Remove(0, 1);

            string[] token = fullurl.Split('/');
            if (token.Length > 0)
            {
                return siteUrl + "/" + token[0];
            }
            return null;
        }

        /// <summary>
        /// Get relative url of file or folder within document library
        /// </summary>
        /// <param name="fullUrl">absolute url passed in</param>
        /// <param name="docLibUrl">absolute url of document library passed in</param>
        /// <returns>relative url of file/folder within document library</returns>
        public string GetRelativeFileFolderName(string fullUrl, string docLibUrl)
        {
            fullUrl = fullUrl.ToLowerInvariant().Replace(docLibUrl.ToLowerInvariant(), "").Trim();
            if (fullUrl.StartsWith("/")) fullUrl = fullUrl.Remove(0, 1);

            return fullUrl;
        }

    }
}
```

SpoAuthUtility.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Text;
using System.Threading.Tasks;
using System.Xml;
using System.Xml.XPath;
using System.Threading;
using System.Runtime.Serialization.Json;
using System.Web;


using Microsoft.Xrm.Sdk;

namespace Plugin_SPConnector
{
    internal class SamlSecurityToken
    {
        public byte[] Token
        {
            get;
            set;
        }

        public DateTime Expires
        {
            get;
            set;
```

```
        }
    }

    internal class SPOAuthCookies
    {
        public string FedAuth
        {
            get;
            set;
        }

        public string RtFA
        {
            get;
            set;
        }

        public Uri Host
        {
            get;
            set;
        }

        public DateTime Expires
        {
            get;
            set;
        }
    }

    public class SpoAuthUtility
    {
        Uri spSiteUrl;
        string username;
        string password;
        Uri adfsIntegratedAuthUrl;
        Uri adfsAuthUrl;
        bool useIntegratedWindowsAuth;
        static SpoAuthUtility current;
        CookieContainer cookieContainer;
        SamlSecurityToken stsAuthToken;


        const string msoStsUrl = "https://login.microsoftonline.com/extSTS.srf";
        const string msoLoginUrl = "https://login.microsoftonline.com/login.srf";
        const string msoHrdUrl = "https://login.microsoftonline.com/GetUserRealm.srf";
        const string wsse = "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-
1.0.xsd";
        const string wsu = "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd";
        const string wst = "http://schemas.xmlsoap.org/ws/2005/02/trust";
        const string saml = "urn:oasis:names:tc:SAML:1.0:assertion";
        const string spowssigninUri = "_forms/default.aspx?wa=wsignin1.0";
        const string contextInfoQuery = "_api/contextinfo";

        public static bool Create(Uri spSiteUrl, string username, string password, bool
useIntegratedWindowsAuth)
        {


            var utility = new SpoAuthUtility(spSiteUrl, username, password, useIntegratedWindowsAuth);
            if (!useIntegratedWindowsAuth)
```

```
            {
                CookieContainer cc = utility.GetCookieContainer();
                var cookies = from Cookie c in cc.GetCookies(spSiteUrl) where c.Name == "FedAuth" select c;
                if (cookies.Count() > 0)
                {
                    current = utility;
                    return true;
                }
                else
                    throw new Exception("Could not retrieve Auth cookies");
            }
            else
            {
                current = utility;
                return true;
            }
        }

        private SpoAuthUtility(Uri spSiteUrl, string username, string password, bool useIntegratedWindowsAuth)
        {
            this.spSiteUrl = spSiteUrl;
            this.username = username;
            this.password = password;
            this.useIntegratedWindowsAuth = useIntegratedWindowsAuth;

            stsAuthToken = new SamlSecurityToken();
        }


        public bool UseIntegratedSecurity { get { return useIntegratedWindowsAuth; } }

        public bool IsUserSet { get { return !string.IsNullOrEmpty(username) && !string.IsNullOrEmpty(password); } }

        public NetworkCredential WindowsCredentials { get { return new NetworkCredential(username, password); } }


        public static SpoAuthUtility Current
        {
            get
            {
                return current;
            }
        }




        public Uri SiteUrl
        {
            get
            {
                return this.spSiteUrl;
            }
        }

        /// <summary>
        /// The method will request the SP ContextInfo and return its FormDigestValue as a String
        /// The FormDigestValue is a second layer of authentication required for several REST queries
        /// </summary>
```

```
        /// <returns>FormDigestValue</returns>
        public static string GetRequestDigest()
        {
            string digest = String.Empty;
            Uri url = new Uri(String.Format("{0}/{1}", SpoAuthUtility.Current.SiteUrl, contextInfoQuery));
            byte[] result = HttpHelper.SendODataJsonRequest(
              url,
              "POST", // Retrieving the Context Info needs a POST Method
              new byte[0],
              (HttpWebRequest)HttpWebRequest.Create(url),
              SpoAuthUtility.Current // pass in the helper object that allows us to make authenticated calls to SPO
rest services
              );

            // use the DataContractJsonSerializer instead of the JavascriptSerializer (system.web.extension.dll
cannot be used in the sandbox)
            MemoryStream stream = new MemoryStream(result);
            stream.Position = 0;

            DataContractJsonSerializer json = new DataContractJsonSerializer(typeof(ContextInfo));
            ContextInfo ci = (ContextInfo)json.ReadObject(stream);

            //ContextInfo ci = (ContextInfo)json.Deserialize(Encoding.UTF8.GetString(result, 0, result.Length),
typeof(ContextInfo));
            digest = (ci != null) ? ci.FormDigestValue : String.Empty;

            return digest;
        }

        /// <summary>
        /// Helper class to handle ContextInfo JSON Deserialisation
        /// </summary>
        class ContextInfo
        {
            public Dictionary<string, Dictionary<string, object>> d { get; set; }
            public String FormDigestValue
            {
                get
                {
                    string value = String.Empty;

                    try
                    {
                        if (d != null && d.FirstOrDefault().Value != null)
                        {
                            value = Convert.ToString(d.FirstOrDefault().Value["FormDigestValue"]);
                        }
                    }
                    catch { }

                    return value;
                }
            }
        }


        public CookieContainer GetCookieContainer()
        {
            if (stsAuthToken != null)
            {
                if (DateTime.Now > stsAuthToken.Expires)
                {
```

```
                    this.stsAuthToken = GetMsoStsSAMLToken();

                    // Check if we have a auth token
                    if (this.stsAuthToken != null)
                    {
                        SPOAuthCookies cookies = GetSPOAuthCookies(this.stsAuthToken);
                        CookieContainer cc = new CookieContainer();

                        Cookie samlAuthCookie = new Cookie("FedAuth", cookies.FedAuth)
                        {
                            Path = "/",
                            Expires = this.stsAuthToken.Expires,
                            Secure = cookies.Host.Scheme.Equals("https"),
                            HttpOnly = true,
                            Domain = cookies.Host.Host
                        };

                        cc.Add(this.spSiteUrl, samlAuthCookie);

                        Cookie rtFACookie = new Cookie("rtFA", cookies.RtFA)
                        {
                            Path = "/",
                            Expires = this.stsAuthToken.Expires,
                            Secure = cookies.Host.Scheme.Equals("https"),
                            HttpOnly = true,
                            Domain = cookies.Host.Host
                        };

                        cc.Add(this.spSiteUrl, rtFACookie);

                        this.cookieContainer = cc;
                    }
                }
            }

            return this.cookieContainer;
        }

        private SPOAuthCookies GetSPOAuthCookies(SamlSecurityToken stsToken)
        {
            // signs in to SPO with the security token issued by MSO STS and gets the fed auth cookies
            // the fed auth cookie needs to be attached to all SPO REST services requests
            SPOAuthCookies spoAuthCookies = new SPOAuthCookies();

            Uri siteUri = this.spSiteUrl;
            Uri wsSigninUrl = new Uri(String.Format("{0}://{1}/{2}", siteUri.Scheme, siteUri.Authority,
spowssigninUri));

            HttpWebRequest request = (HttpWebRequest)HttpWebRequest.Create(wsSigninUrl);
            request.CookieContainer = new CookieContainer();

            byte[] responseData = HttpHelper.SendHttpRequest(
                wsSigninUrl,
                "POST",
                stsToken.Token,
                "application/x-www-form-urlencoded",
                request,
                null);

            if (request != null && responseData != null)
            {
                using (HttpWebResponse response = (HttpWebResponse)request.GetResponse())
```

```
                {
                    spoAuthCookies.FedAuth = (response.Cookies["FedAuth"] != null) ? response.Cookies["FedAuth"].Value : null;
                    spoAuthCookies.RtFA = (response.Cookies["rtFA"] != null) ? response.Cookies["rtFA"].Value : null;
                    spoAuthCookies.Expires = stsToken.Expires;
                    spoAuthCookies.Host = wsSigninUrl;
                }
            }

            return spoAuthCookies;
        }

        private Uri GetAdfsAuthUrl()
        {
            Uri corpAdfsProxyUrl = null;

            Uri msoHrdUri = new Uri(msoHrdUrl);
            HttpWebRequest request = (HttpWebRequest)HttpWebRequest.Create(msoHrdUri);
            // make a post request with the user's login name to
            // MSO HRD (Home Realm Discovery) service so it can find
            // out the url of the federation service (corporate ADFS)
            // responsible for authenticating the user

            byte[] response = HttpHelper.SendHttpRequest(
                new Uri(msoHrdUrl),
                "POST",
                Encoding.UTF8.GetBytes(String.Format("handler=1&login={0}", this.username)), // pass in the login name in the body of the form
                "application/x-www-form-urlencoded",
                request);

            // convert byte[] to string
            STSInfo info = new STSInfo();
            if (response != null)
            {
                string s = GetTokenFromJson(UTF8Encoding.UTF8.GetString(response), "AuthURL");
                if (!string.IsNullOrEmpty(s)) info.AuthURL = s;
            }

            if (info != null && !String.IsNullOrEmpty(info.AuthURL))
            {
                corpAdfsProxyUrl = new Uri(info.AuthURL);
            }


            return corpAdfsProxyUrl;
        }

        /// <summary>
        /// Some JSON objects cannot be deserialized.
        /// Then we do this by hand
        /// </summary>
        /// <param name="json">json response</param>
        /// <param name="tokenName">name of the token</param>
        /// <returns>value of token</returns>
        private string GetTokenFromJson(string json, string tokenName)
        {
            tokenName = "\"" + tokenName + "\":";
            if (json != null)
            {
                string[] tokens = json.Split(',');
```

```
            foreach (string token in tokens)
            {
                if (token.Trim().StartsWith(tokenName, StringComparison.InvariantCultureIgnoreCase))
                {
                    return token.Replace(tokenName, "").Replace("\"", "").Trim();
                }
            }
        }
        return null;
    }


    /// <summary>
    /// Helper class to handle STS Info JSON Deserialisation
    /// </summary>
    [Serializable]
    class STSInfo
    {
        public String AuthURL { get; set; }
    }



    private string GetAdfsSAMLTokenUsernamePassword()
    {
        // makes a seurity token request to the corporate ADFS proxy usernamemixed endpoint using
        // the user's corporate credentials. The logon token is used to talk to MSO STS to get
        // an O365 service token that can then be used to sign into SPO.
        string samlAssertion = null;

        // the corporate ADFS proxy endpoint that issues SAML seurity tokens given username/password credentials
        string stsUsernameMixedUrl = String.Format("https://{0}/adfs/services/trust/2005/usernamemixed/", adfsAuthUrl.Host);

        // generate the WS-Trust security token request SOAP message passing in the user's corporate credentials
        // and the site we want access to. We send the token request to the corporate ADFS proxy usernamemixed endpoint.
        byte[] requestBody = Encoding.UTF8.GetBytes(ParameterizeSoapRequestTokenMsgWithUsernamePassword(
            "urn:federation:MicrosoftOnline", // we are requesting a logon token to talk to the Microsoft Federation Gateway
            this.username,
            this.password,
            stsUsernameMixedUrl));

        try
        {
            Uri stsUrl = new Uri(stsUsernameMixedUrl);
            HttpWebRequest request = (HttpWebRequest)HttpWebRequest.Create(stsUrl);

            byte[] responseData = HttpHelper.SendHttpRequest(
                stsUrl,
                "POST",
                requestBody,
                "application/soap+xml; charset=utf-8",
                request,
                null);

            if (responseData != null)
            {
```

```
                    StreamReader sr = new StreamReader(new MemoryStream(responseData), Encoding.GetEncoding("utf-8"));
                    XPathNavigator nav = new XPathDocument(sr).CreateNavigator();
                    XmlNamespaceManager nsMgr = new XmlNamespaceManager(nav.NameTable);
                    nsMgr.AddNamespace("t", "http://schemas.xmlsoap.org/ws/2005/02/trust");
                    XPathNavigator requestedSecurityToken = nav.SelectSingleNode("//t:RequestedSecurityToken", nsMgr);

                    // Ensure whitespace is reserved
                    XmlDocument doc = new XmlDocument();
                    doc.LoadXml(requestedSecurityToken.InnerXml);
                    doc.PreserveWhitespace = true;
                    samlAssertion = doc.InnerXml;
                }

            }
            catch
            {
                // we failed to sign the user using corporate credentials

            }

            return samlAssertion;
        }

        private string GetAdfsSAMLTokenWinAuth()
        {
            // makes a seurity token request to the corporate ADFS proxy integrated auth endpoint.
            // If the user is logged on to a machine joined to the corporate domain with her Windows credentials and connected
            // to the corporate network Kerberos automatically takes care of authenticating the security token
            // request to ADFS.
            // The logon token is used to talk to MSO STS to get an O365 service token that can then be used to sign into SPO.

            string samlAssertion = null;

            HttpWebRequest request = (HttpWebRequest)HttpWebRequest.Create(this.adfsIntegratedAuthUrl);
            request.UseDefaultCredentials = true; // use the default credentials so Kerberos can take care of authenticating our request

            byte[] responseData = HttpHelper.SendHttpRequest(
                this.adfsIntegratedAuthUrl,
                "GET",
                null,
                "text/html; charset=utf-8",
                request);


            if (responseData != null)
            {
                try
                {
                    StreamReader sr = new StreamReader(new MemoryStream(responseData), Encoding.GetEncoding("utf-8"));
                    XPathNavigator nav = new XPathDocument(sr).CreateNavigator();
                    XPathNavigator wresult = nav.SelectSingleNode("/html/body/form/input[@name='wresult']");
                    if (wresult != null)
                    {
                        string RequestSecurityTokenResponseText = wresult.GetAttribute("value", "");
```

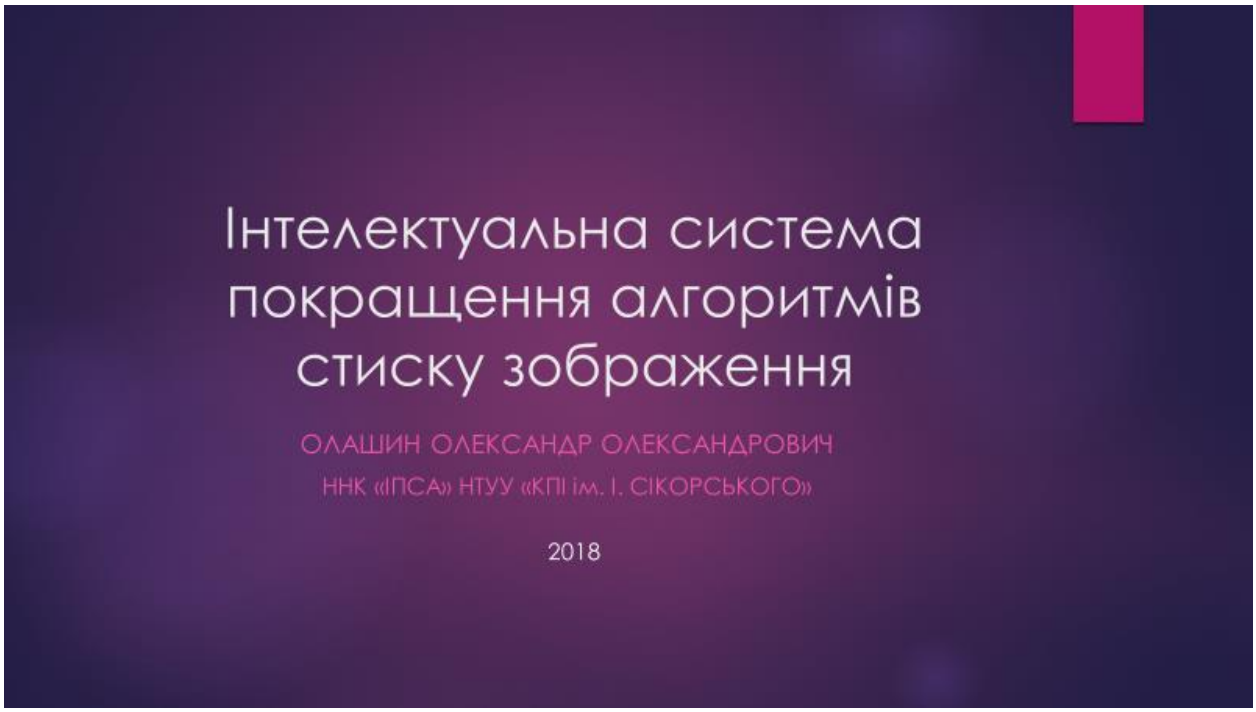
```
                    sr = new StreamReader(new MemoryStream(Encoding.UTF8.GetBytes(RequestSecurityTokenResponseText)));
                    nav = new XPathDocument(sr).CreateNavigator();
                    XmlNamespaceManager nsMgr = new XmlNamespaceManager(nav.NameTable);
                    nsMgr.AddNamespace("t", "http://schemas.xmlsoap.org/ws/2005/02/trust");
                    XPathNavigator requestedSecurityToken = nav.SelectSingleNode("//t:RequestedSecurityToken", nsMgr);

                    // Ensure whitespace is reserved
                    XmlDocument doc = new XmlDocument();
                    doc.LoadXml(requestedSecurityToken.InnerXml);
                    doc.PreserveWhitespace = true;
                    samlAssertion = doc.InnerXml;
                  }

                }
                catch
                {
                  // we failed to sign the user using integrated Windows Auth

                }
              }


              return samlAssertion;

            }


            private SamlSecurityToken GetMsoStsSAMLToken()
            {
              // Makes a request that conforms with the WS-Trust standard to
              // Microsoft Online Services Security Token Service to get a SAML
              // security token back so we can then use it to sign the user to SPO

              SamlSecurityToken samlST = new SamlSecurityToken();
              byte[] saml11RTBytes = null;
              string logonToken = null;

              // find out whether the user's domain is a federated domain
              this.adfsAuthUrl = GetAdfsAuthUrl();

              // get logon token using windows integrated auth when the user is connected to the corporate network
              if (this.adfsAuthUrl != null && this.useIntegratedWindowsAuth)
              {
                UriBuilder ub = new UriBuilder();
                ub.Scheme = this.adfsAuthUrl.Scheme;
                ub.Host = this.adfsAuthUrl.Host;
                ub.Path = string.Format("{0}auth/integrated/", this.adfsAuthUrl.LocalPath);

                // specify in the query string we want a logon token to present to the Microsoft Federation Gateway
                // for the corresponding user
                ub.Query = String.Format("{0}&wa=wsignin1.0&wtrealm=urn:federation:MicrosoftOnline", this.adfsAuthUrl.Query.Remove(0, 1)).
                  Replace("&username=", String.Format("&username={0}", this.username));

                this.adfsIntegratedAuthUrl = ub.Uri;

                // get the logon token from the corporate ADFS using Windows Integrated Auth
                logonToken = GetAdfsSAMLTokenWinAuth();

                if (!string.IsNullOrEmpty(logonToken))
```

```
                {
                    // generate the WS-Trust security token request SOAP message passing in the logon token we got from the corporate ADFS
                    // and the site we want access to
                    saml11RTBytes = Encoding.UTF8.GetBytes(ParameterizeSoapRequestTokenMsgWithAssertion(
                        this.spSiteUrl.ToString(),
                        logonToken,
                        msoStsUrl));
                }
            }

            // get logon token using the user's corporate credentials. Likely when not connected to the corporate network
            if (logonToken == null && this.adfsAuthUrl != null && !string.IsNullOrEmpty(password))
            {
                logonToken = GetAdfsSAMLTokenUsernamePassword(); // get the logon token from the corporate ADFS proxy usernamemixed enpoint

                if (logonToken != null)
                {
                    // generate the WS-Trust security token request SOAP message passing in the logon token we got from the corporate ADFS
                    // and the site we want access to
                    saml11RTBytes = Encoding.UTF8.GetBytes(ParameterizeSoapRequestTokenMsgWithAssertion(
                        this.spSiteUrl.ToString(),
                        logonToken,
                        msoStsUrl));
                }
            }

            if (logonToken == null && this.adfsAuthUrl == null && !string.IsNullOrEmpty(password)) // login with O365 credentials. Not a federated login.
            {
                // generate the WS-Trust security token request SOAP message passing in the user's credentials and the site we want access to
                saml11RTBytes = Encoding.UTF8.GetBytes(ParameterizeSoapRequestTokenMsgWithUsernamePassword(
                    this.spSiteUrl.ToString(),
                    this.username,
                    this.password,
                    msoStsUrl));
            }

            if (saml11RTBytes != null)
            {
                Uri MsoSTSUri = new Uri(msoStsUrl);

                HttpWebRequest request = (HttpWebRequest)HttpWebRequest.Create(MsoSTSUri);


                byte[] responseData = HttpHelper.SendHttpRequest(
                    MsoSTSUri,
                    "POST",
                    saml11RTBytes,
                    "application/soap+xml; charset=utf-8",
                    request,
                    null);

                StreamReader sr = new StreamReader(new MemoryStream(responseData), Encoding.GetEncoding("utf-8"));
```

```
                XPathNavigator nav = new XPathDocument(sr).CreateNavigator();
                XmlNamespaceManager nsMgr = new XmlNamespaceManager(nav.NameTable);
                nsMgr.AddNamespace("wsse", "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd");
                XPathNavigator binarySecurityToken = nav.SelectSingleNode("//wsse:BinarySecurityToken",
nsMgr);

                if (binarySecurityToken != null)
                {
                   string binaryST = binarySecurityToken.InnerXml;

                   nsMgr.AddNamespace("wsu", "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd");
                   XPathNavigator expires = nav.SelectSingleNode("//wsu:Expires", nsMgr);

                   if (!String.IsNullOrEmpty(binarySecurityToken.InnerXml) &&
!String.IsNullOrEmpty(expires.InnerXml))
                   {
                      samlST.Token = Encoding.UTF8.GetBytes(binarySecurityToken.InnerXml);
                      samlST.Expires = DateTime.Parse(expires.InnerXml);
                   }
                }
                else
                {
                   // We didn't get security token
                }
             }

             return samlST;
          }

          private string ParameterizeSoapRequestTokenMsgWithUsernamePassword(string url, string username,
string password, string toUrl)
          {
             System.Text.StringBuilder s = new System.Text.StringBuilder();
             s.Append("<s:Envelope xmlns:s=\"http://www.w3.org/2003/05/soap-envelope\"
xmlns:a=\"http://www.w3.org/2005/08/addressing\" xmlns:u=\"http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd\">");
             s.Append("<s:Header>");
             s.Append("<a:Action
s:mustUnderstand=\"1\">http://schemas.xmlsoap.org/ws/2005/02/trust/RST/Issue</a:Action>");
             s.Append("<a:ReplyTo>");
             s.Append("<a:Address>http://www.w3.org/2005/08/addressing/anonymous</a:Address>");
             s.Append("</a:ReplyTo>");
             s.Append("<a:To s:mustUnderstand=\"1\">[toUrl]</a:To>");
             s.Append("<o:Security s:mustUnderstand=\"1\" xmlns:o=\"http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd\">");
             s.Append("<o:UsernameToken>");
             s.Append("<o:Username>[username]</o:Username>");
             s.Append("<o:Password>[password]</o:Password>");
             s.Append("</o:UsernameToken>");
             s.Append("</o:Security>");
             s.Append("</s:Header>");
             s.Append("<s:Body>");
             s.Append("<t:RequestSecurityToken xmlns:t=\"http://schemas.xmlsoap.org/ws/2005/02/trust\">");
             s.Append("<wsp:AppliesTo xmlns:wsp=\"http://schemas.xmlsoap.org/ws/2004/09/policy\">");
             s.Append("<a:EndpointReference>");
             s.Append("<a:Address>[url]</a:Address>");
             s.Append("</a:EndpointReference>");
             s.Append("</wsp:AppliesTo>");

s.Append("<t:KeyType>http://schemas.xmlsoap.org/ws/2005/05/identity/NoProofKey</t:KeyType>");
```

```
                s.Append("<t:RequestType>http://schemas.xmlsoap.org/ws/2005/02/trust/Issue</t:RequestType>");
                s.Append("<t:TokenType>urn:oasis:names:tc:SAML:1.0:assertion</t:TokenType>");
                s.Append("</t:RequestSecurityToken>");
                s.Append("</s:Body>");
                s.Append("</s:Envelope>");


                string samlRTString = s.ToString();
                samlRTString = samlRTString.Replace("[username]", username);
                samlRTString = samlRTString.Replace("[password]", password);
                samlRTString = samlRTString.Replace("[url]", url);
                samlRTString = samlRTString.Replace("[toUrl]", toUrl);

                return samlRTString;
            }

            private string ParameterizeSoapRequestTokenMsgWithAssertion(string url, string samlAssertion,
string toUrl)
            {
                System.Text.StringBuilder s = new System.Text.StringBuilder();
                s.Append("<s:Envelope xmlns:s=\"http://www.w3.org/2003/05/soap-envelope\"
xmlns:a=\"http://www.w3.org/2005/08/addressing\" xmlns:u=\"http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd\">");
                s.Append("<s:Header>");
                s.Append("<a:Action
s:mustUnderstand=\"1\">http://schemas.xmlsoap.org/ws/2005/02/trust/RST/Issue</a:Action>");
                s.Append("<a:ReplyTo>");
                s.Append("<a:Address>http://www.w3.org/2005/08/addressing/anonymous</a:Address>");
                s.Append("</a:ReplyTo>");
                s.Append("<a:To s:mustUnderstand=\"1\">[toUrl]</a:To>");
                s.Append("<o:Security s:mustUnderstand=\"1\" xmlns:o=\"http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd\">[assertion]");
                s.Append("</o:Security>");
                s.Append("</s:Header>");
                s.Append("<s:Body>");
                s.Append("<t:RequestSecurityToken xmlns:t=\"http://schemas.xmlsoap.org/ws/2005/02/trust\">");
                s.Append("<wsp:AppliesTo xmlns:wsp=\"http://schemas.xmlsoap.org/ws/2004/09/policy\">");
                s.Append("<a:EndpointReference>");
                s.Append("<a:Address>[url]</a:Address>");
                s.Append("</a:EndpointReference>");
                s.Append("</wsp:AppliesTo>");

s.Append("<t:KeyType>http://schemas.xmlsoap.org/ws/2005/05/identity/NoProofKey</t:KeyType>");
                s.Append("<t:RequestType>http://schemas.xmlsoap.org/ws/2005/02/trust/Issue</t:RequestType>");
                s.Append("<t:TokenType>urn:oasis:names:tc:SAML:1.0:assertion</t:TokenType>");
                s.Append("</t:RequestSecurityToken>");
                s.Append("</s:Body>");
                s.Append("</s:Envelope>");

                string samlRTString = s.ToString();
                samlRTString = samlRTString.Replace("[assertion]", samlAssertion);
                samlRTString = samlRTString.Replace("[url]", url);
                samlRTString = samlRTString.Replace("[toUrl]", toUrl);
                return samlRTString;
            }
        }
    }
```

ДОДАТОК Б ІЛЮСТРАТИВНИЙ МАТЕРІАЛ



Мета роботи

3

- Метою даної роботи є розробка інтелектуальної системи покращення алгоритмів стиску зображень на основі нейронних мереж.

Актуальність роботи

4

- Сучасні стандарти вимагають ефективних алгоритмів стиску даних і просто змінити вже існуючі формати на нові є не простим завданням. Тому створення сервісу для покращення роботи алгоритмів без зміни самих алгоритмів є актуальним завданням.

Постановка задачі магістерської дисертації

5

- Проаналізувати існуючі методи стиску зображень.
- Проаналізувати існуючі реалізації покращення стиску зображень.
- Розробити програмний продукт, що реалізує систему інтелектуального покращення алгоритмів стиску зображень.

Аналіз існуючих методів

6

- Основні напрями досліджень:
  - Створення нових алгоритмів стиску зображень
  - Модифікація вже існуючих
  - Покращення систем роботи алгоритмів

Запропонований метод

7

- Метод обраний в цій роботі являє собою систему, що складається з двох згорткових нейронних мереж Compact CNN та Reconstractive CNN. Перша нейронна мережа створює зменшену версію зображення, яка подається на кодек*. Потім для отримання вихідного зображення його декодований варіант подається на другу нейронну мережу.

*Під кодеком розуміють певний стандартний алгоритм стиску (в даній роботі було використано JPEG)

Архітектура системи

8

Вибір платформи та мови програмування

9

- Як основну платформу було обрано Azure Machine Learning Studio. Як основну мову було обрано Net#

Azure
Machine Learning

Результат роботи

10

- Для порівняння було обрано два основні показники – коефіцієнт пікового сигналу до шумового співвідношення (PSNR) та індекс структурної схожості (SSIM).
- Для тестування використовувались 5 класичних зображень, які застосовуються для тестування.

Результати роботи за критерієм PSNR (dB)

11

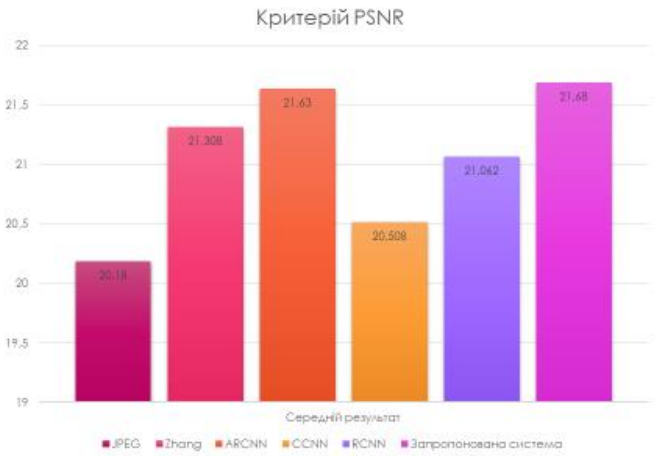
Метод	Метелик	Дім	Папуга	Листя	Фотограф	Середній результат
JPEG	22.58	27.77	26.19	22.49	24.45	20.18
Zhang	24.20	29.24	27.78	24.13	25.39	21.308
ARCNN	25.64	29.68	28.13	25.07	25.27	21.63
CCNN	22.85	27.9	27.01	22.85	24.78	20.508
RCNN	24.55	28.34	27.57	24.23	25.17	21.062
Запропонована система	**25.78**	**29.78**	**28.01**	**25.03**	**25.58**	**21.68**

*більше краще

Аналіз результатів

12

За критерієм PSNR запропонована система показала найвищий середній результат 21.68, перегнавши ARCNN, результат якої - 21.63.



Результати роботи за критерієм SSIM

13

Метод	Метелик	Дім	Папуга	Листя	Фотограф	Середній результат
JPEG	0.7378	0.7733	0.7581	0.7775	0.7283	0.60744
Zhang	0.8313	0.8141	0.8308	0.8548	0.7672	0.65338
ARCNN	0.8741	0.8209	0.8446	0.8983	0.7674	0.66624
CCNN	0.7201	0.8019	0.8232	0.7761	0.7582	0.63188
RCNN	0.867	0.8189	0.8411	0.8643	0.7543	0.65572
Запропонована система	**0.8732**	**0.8321**	**0.8367**	**0.8821**	**0.7677**	**0.66372**

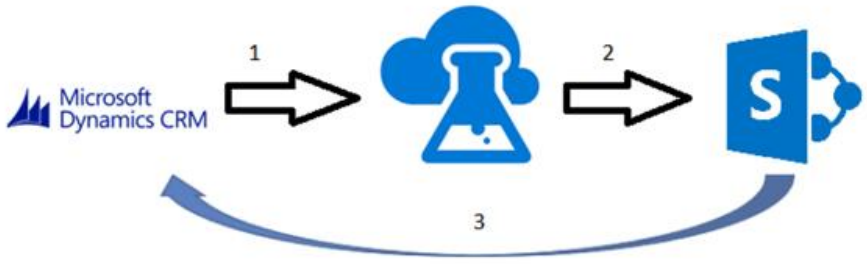
*ближче до 1 краще

Аналіз результатів

14

За критерієм SSIM розроблена мережа показала другий середній результат серед протестованих систем (0.66372 проти 0.66624 ARCNN).

15



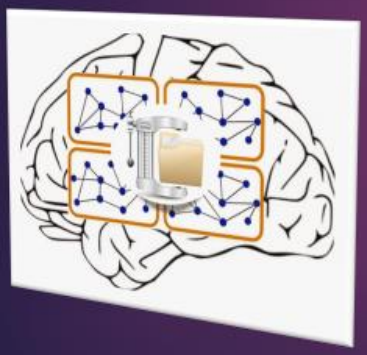
Стартап-проект: «Інтеграційний пакет для MS D365 CE та MS Sharepoint»

Переваги

16

№ п/п	Техніко-економічні характеристики ідеї	Конкуренти			
		Мій проект	PowerObject	Magnetic One	Initial integration
1.	Швидкість роботи	+	+	+	+
2.	Легкість використання	+	+	+/-	-
3.	Стиск даних	+	-	-	-
4.	Ціна	+	-	-	+/-

Висновки



17

Новизна роботи:

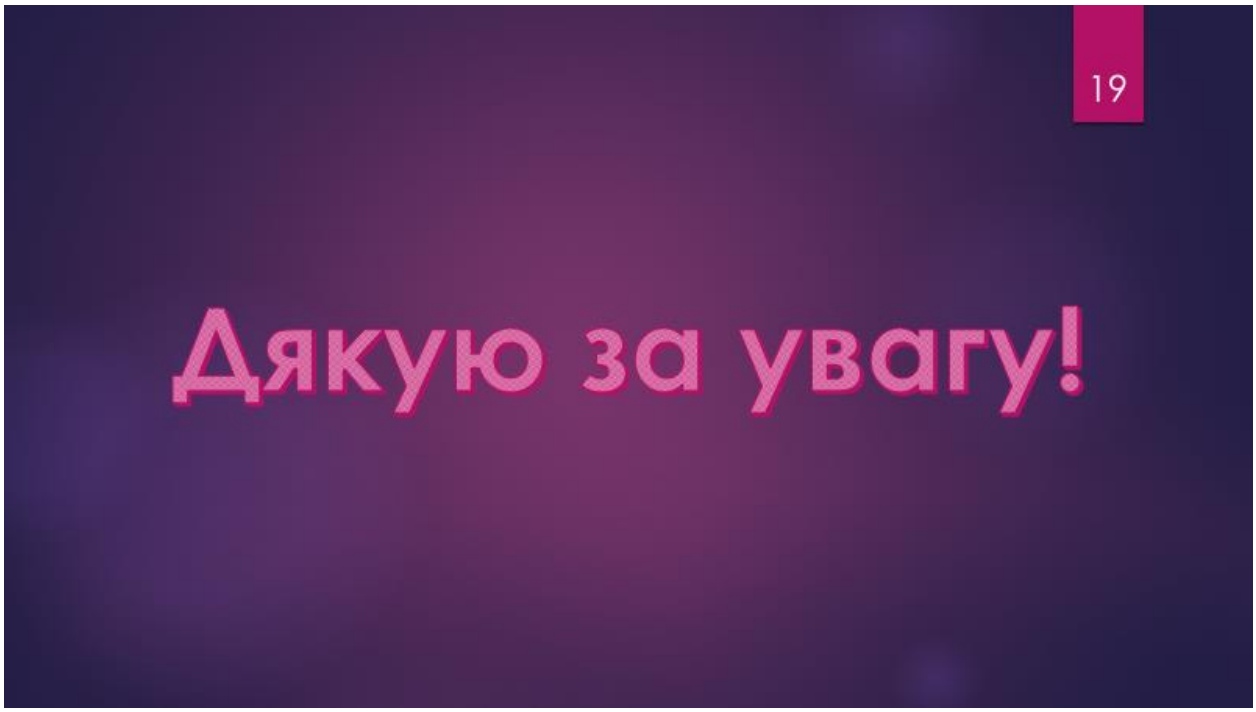
- запропоновано принципово нову систему покращення алгоритмів стиску зображень;
- обґрунтовано використання власних підходів до розв'язку задачі стиску зображень.

Висновки

18

Практична цінність роботи:

- Розроблена система може використовуватись для покращення зображень, а реалізація даної системи, як REST веб сервісу дозволяє легко застосовувати її як елемент більш складної системи.
- Розроблена система в порівнянні з однією з найкращих систем (ARCNN) показала себе з найкращого боку
- Реалізовано стартап – проект інтеграційний пакет MS Dynamics CRM та MS Sharepoint, з використання розробленої системи покращення зображення.


19
Дякую за увагу!

ДОДАТОК В СПИСОК НАУКОВИХ ПУБЛІКАЦІЙ

Статті

1. Олашин О. О. Система покращення алгоритмів стиску зображення // Міжнародний науковий журнал "Інтернаука". — 2018. — №9.