Першим завданням лабораторної роботи було виконання скрипту, який було наведено в прикладі 1 (Див. Рис. 2.1). Після виконання скрипту, було успішно створено нейронну мережу з використанням шарів Flatten, Dense та функцій активації (activation). Матриця розбіжностей для цього виконання показана на Рис. 2.3, а графіки можна побачити на Рис. 2.2.

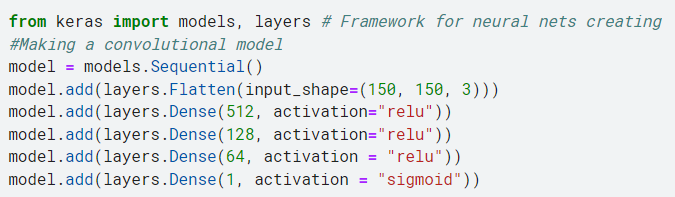


Рисунок 2.1 – Скріншот програми для створення згорткової моделі

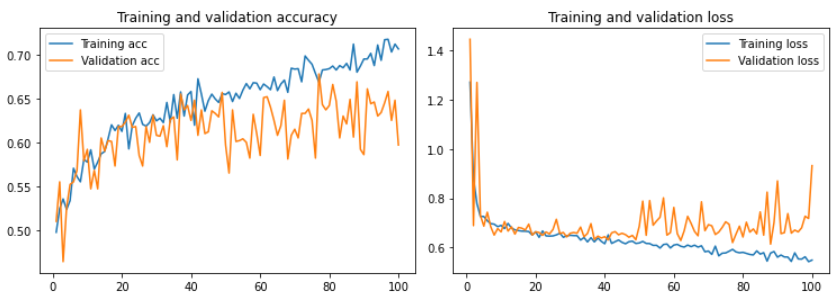


Рисунок 2.2 – Графіки точності тренування та перевірки і втрат тренування та перевірки відповідно

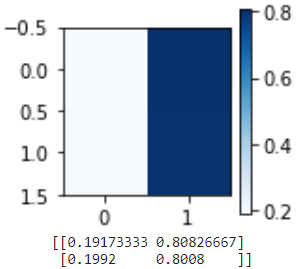


Рисунок 2.3 – Матриця розбіжностей (confusion matrix)

У другому завданні потрібно було дослідити документацію по шару Dense. В результаті стають зрозумілими операції з прикладу 1 (Див. Рис. 2.1). У першому рядку програма імпортує класи models і layers з бібліотеки Keras, які будуть використовуватись у наступному коді. Наступний рядок створює модель нейронної мережі, яка буде використовуватись для навчання і передбачення. Використовується клас Sequential, який дозволяє визначити модель шляхом послідовного додавання шарів. Третій рядок додає перший шар до моделі. Використовується клас Flatten, який призначений для перетворення вхідних даних з багатовимірної до одновимірної форми. Вхідна форма цього шару вказується аргументом input\_shape=(150, 150, 3), це означає, що очікується вхід з розмірністю 150х150 пікселів та 3 канали кольору (RGB). Наступні рядки також додають шари класу Dense до моделі, цей клас представляє повнозв'язний шар нейронної мережі. Перші аргументи (512, 128, 64 та 1) вказують на кількість нейронів у відповідному шарі, а другі аргументи вказують на тип активаційної функції. Для перших трьох шарів це функція relu (англ. rectified linear unit), її зображено на Рис. 2.4. Для останнього шару це функція sigmoid (Див. Рис. 2.5), яка використовується для бінарної класифікації.

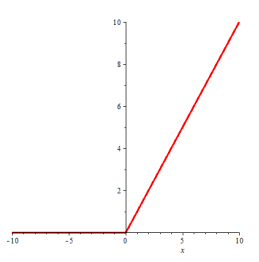


Рисунок 2.4 – Функція активації relu

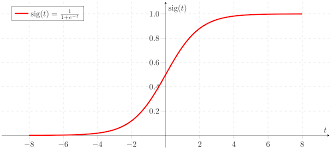


Рисунок 2.5 – Функція активації sigmoid

У останньому завданні цієї роботи потрібно було провести обчислювальні експерименти з нейронною мережею. А саме змінювати кількість шарів, кількість нейронів в шарах, види функцій активації. Для зручності показу матриці було змінено її вигляд з використанням бібліотеки seaborn, скріншоти коду можна побачити на Рис. 2.6.

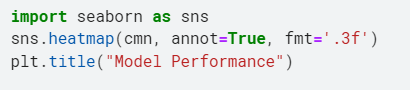


Рисунок 2.6 – Скріншот коду відображення матриці розбіжностей

Як видно з результатів першого запуску (Див. Рис. 2.2, Рис. 2.3), нейронна мережа є перенавченою так як за матрицею розбіжностей видно, що один клас прогнозується дуже добре, а інший майже не прогнозується. Також з кривої validation loss бачимо що графік стає нестабільним приблизна з 50 епохи, тож зменшимо кількість епох до цього значення. Тепер, щоб покращити результати роботи нашої нейромережі спробуємо для початку додати ще один шар з 32 нейронами (Див. Рис. 2.7). Результати тренування показані на Рис. 2.8 та Рис. 2.9, на першому з них видні незначні зміни, але система все ще є неточною і видає майже випадкові передбачення.

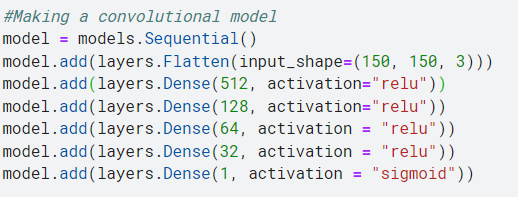


Рисунок 2.7 – Скріншот зміненого коду, доданий шар з 32 нейронами

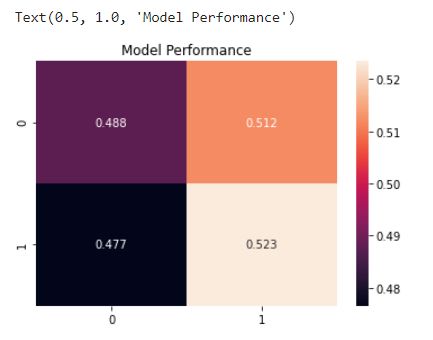


Рисунок 2.8 – Матриця розбіжностей (confusion matrix)

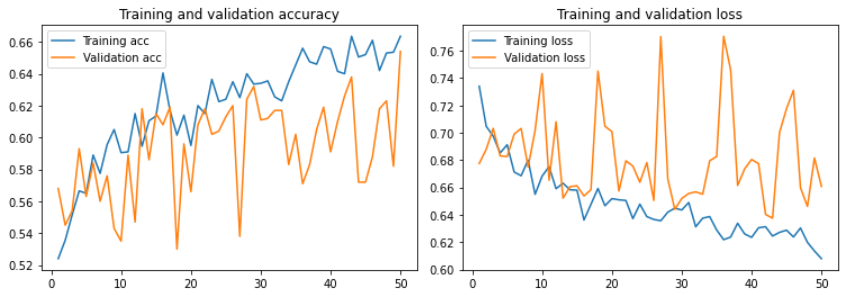


Рисунок 2.9 – Графіки точності тренування та перевірки і втрат тренування та перевірки відповідно

Спробуємо додати ще один шар (Див. Рис. 2.10), результати показано на Рис. 2.11 та Рис. 2.12. На графіках бачимо, що у порівнянні з попереднім варіантом стабілізувалась валідаційна точність та втрати, проте не було отримано значних покращень у точності моделі відповідно до матриці.

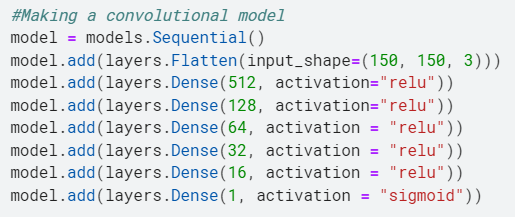


Рисунок 2.10 – Скріншот зміненого коду, доданий шар з 16 нейронами

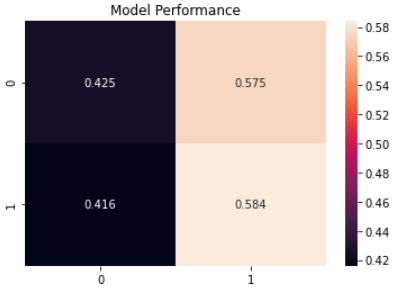


Рисунок 2.11 – Матриця розбіжностей (confusion matrix)

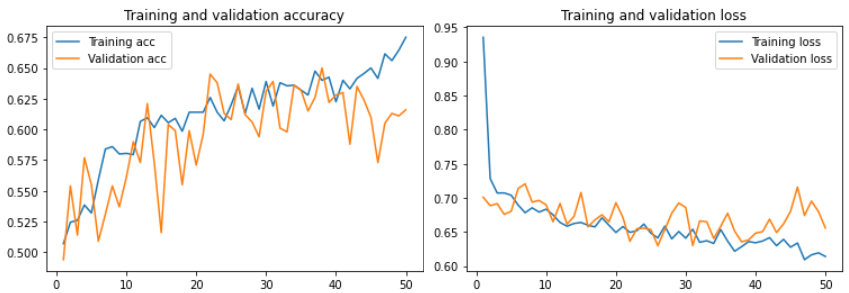


Рисунок 2.12 – Графіки точності тренування та перевірки і втрат тренування та перевірки відповідно

Тепер спробуємо зменшити кількість шарів до чотирьох (Див. Рис. 2.13), результати роботи такої мережі показано на Рис. 2.14 та Рис. 2.15. Для такої нейромережі точність ще більше погіршилась, як бачимо з матриці розбіжності. Також, в порівнянні з графіками попереднього варіанта стабільність погіршилась.

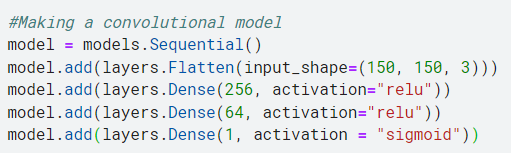


Рисунок 2.13 – Скріншот зміненого коду, мережа з чотирма шарами

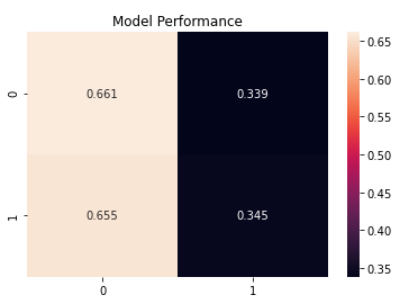


Рисунок 2.14 – Матриця розбіжностей (confusion matrix)

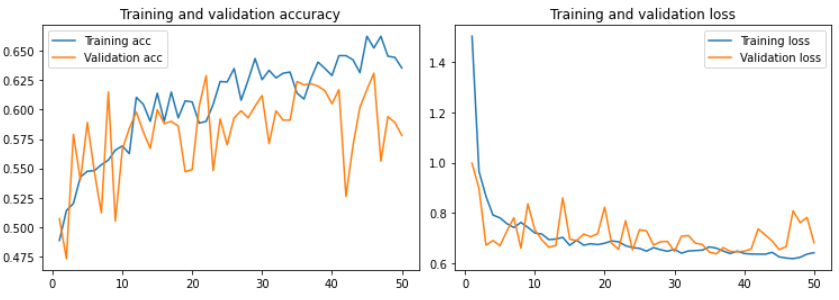


Рисунок 2.15 – Графіки точності тренування та перевірки і втрат тренування та перевірки відповідно

Для нейронної мережі з семи шарів зменшити кількість нейронів деяких шарів Див. Рис. 2.16. Результати показані на Рис. 2.17.та Рис. 2.18. Якщо порівняти матриці розбіжностей Рис. 2.11 та Рис. 2.17 побачимо, що у другому випадку результати погіршились. Також при порівнянні графіків (Рис. 2.12 та Рис. 2.18) бачимо, що стабільність погіршилась у другому випадку.

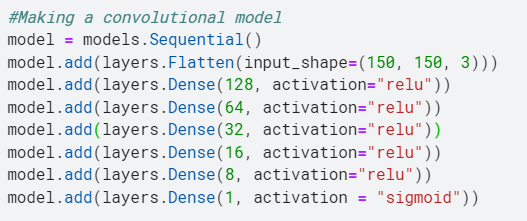


Рисунок 2.16 – Скріншот зміненого коду, мережа з семи шарів та зміненою кількістю нейронів

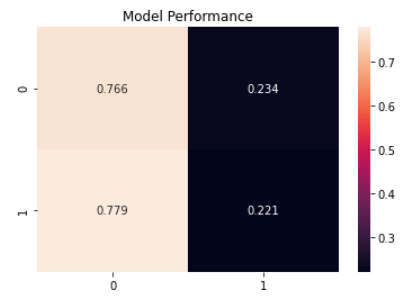


Рисунок 2.17 – Матриця розбіжностей (confusion matrix)



Рисунок 2.18 – Графіки точності тренування та перевірки і втрат тренування та перевірки відповідно

Змінимо активаційну функцію relu спочатку на selu (Див. Рис. 2.19), а потім на softplus (2.22), результати показані на Рис. 2.20, Рис. 2.21 та Рис. 2.23, Рис. 2.24 відповідно. Також протестуємо тренування нейромережі з використанням активаційної функції elu (Див. Рис. 2.25, Рис. 2.26, Рис. 2.27) При порівнянні графіків функцій активацій relu, selu, softplus та elu бачимо, що найбільш плавний графік має мережа натренована з використанням активаційної функції selu.

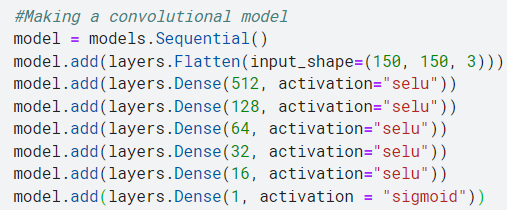


Рисунок 2.19 – Скріншот зміненого коду, мережа з семи шарів та зміненою активаційною функцією selu

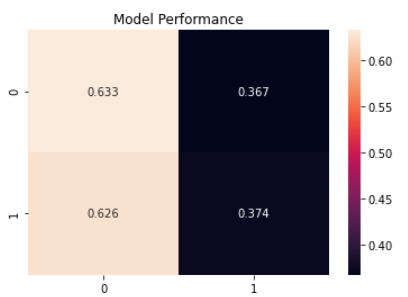


Рисунок 2.20 – Матриця розбіжностей (confusion matrix)

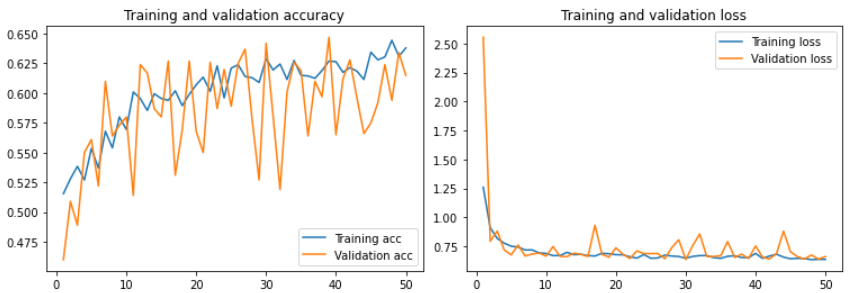


Рисунок 2.21– Графіки точності тренування та перевірки і втрат тренування та перевірки відповідно

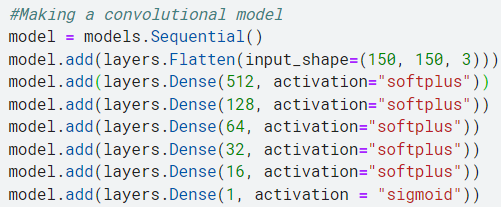


Рисунок 2.22 – Скріншот зміненого коду, мережа з семи шарів та зміненою активаційною функцією softplus

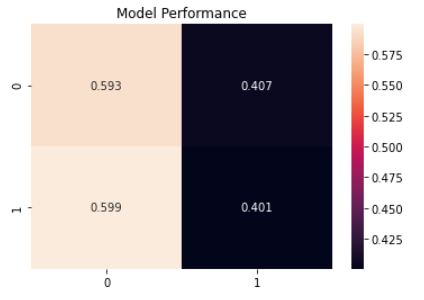


Рисунок 2.23 – Матриця розбіжностей (confusion matrix)

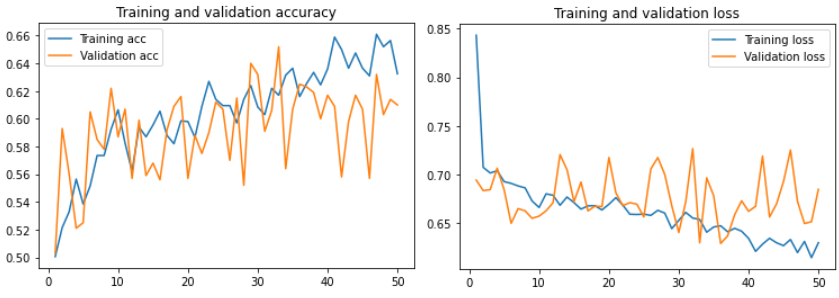


Рисунок 2.24– Графіки точності тренування та перевірки і втрат тренування та перевірки відповідно

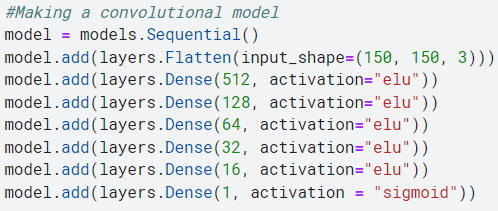


Рисунок 2.25 – Скріншот зміненого коду, мережа з семи шарів та зміненою активаційною функцією elu

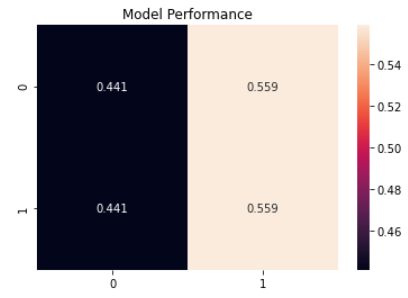


Рисунок 2.26 – Матриця розбіжностей (confusion matrix)

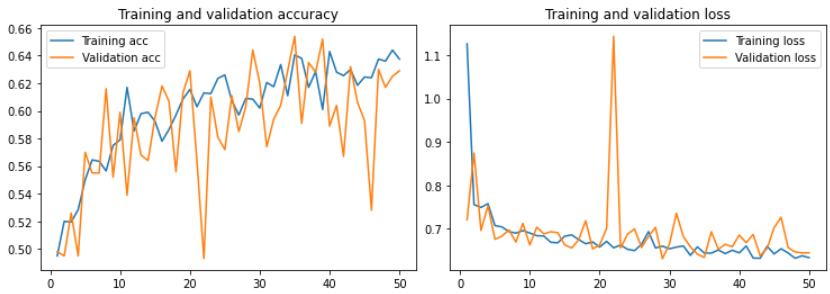


Рисунок 2.27 – Графіки точності тренування та перевірки і втрат тренування та перевірки відповідно

Спробуємо тепер збільшити кількість шарів до 8 (Див. Рис. 2.25), результати роботи такої нейромережі відображені на Рис. 2.26 та Рис. 2.27. Хоча матриця розбіжностей все ще є неточною, графік став більш стабільним у порівнянні з усіма іншими версіями.

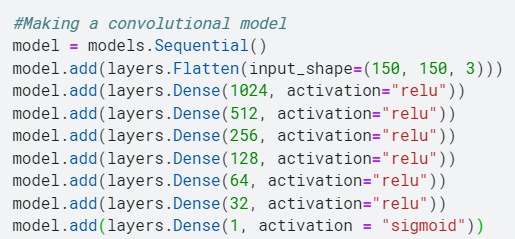


Рисунок 2.28 – Скріншот зміненого коду, 8 шарів у нейромережі

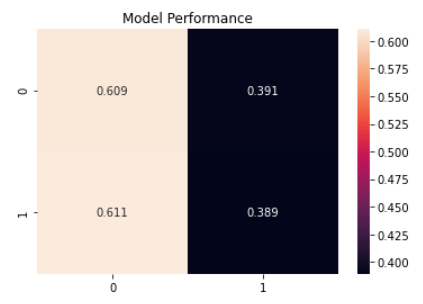


Рисунок 2.29 – Матриця розбіжностей (confusion matrix)

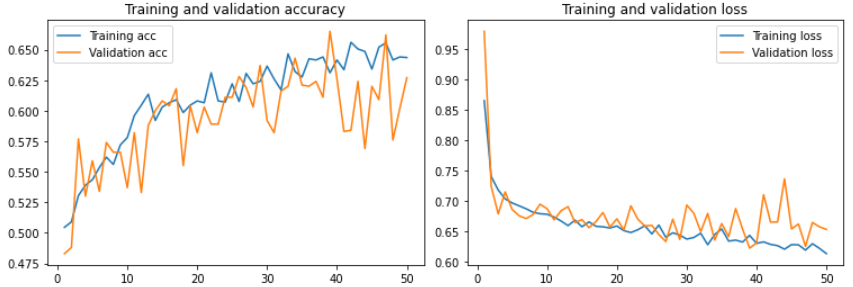


Рисунок 2.30– Графіки точності тренування та перевірки і втрат тренування та перевірки відповідно

Тепер збільшимо кількість нейронів і використаємо активаційну функцію selu (Див. Рис. 2.31). Результати показані на Рис. 2.32 та Рис. 2.33. На графіку втрат тренування та перевірки видно що крива є більш стабільною ніж в інших випадках, але значення матриці розбіжностей все ще вказують на погану прогнозованість.

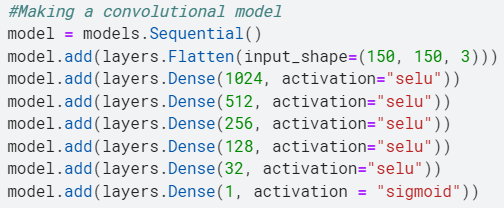


Рисунок 2.31 – Скріншот зміненого коду

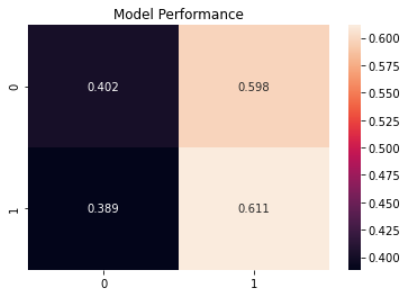


Рисунок 2.29 – Матриця розбіжностей (confusion matrix)

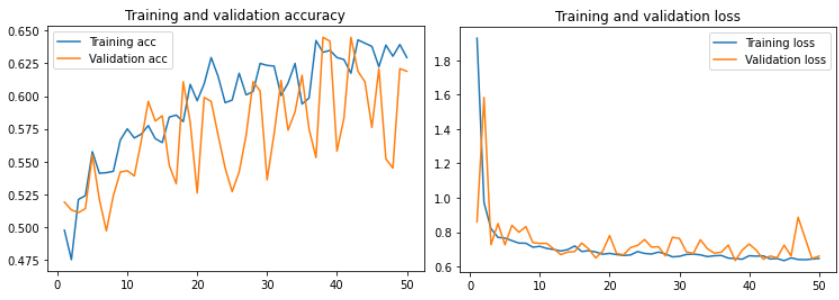


Рисунок 2.30– Графіки точності тренування та перевірки і втрат тренування та перевірки відповідно