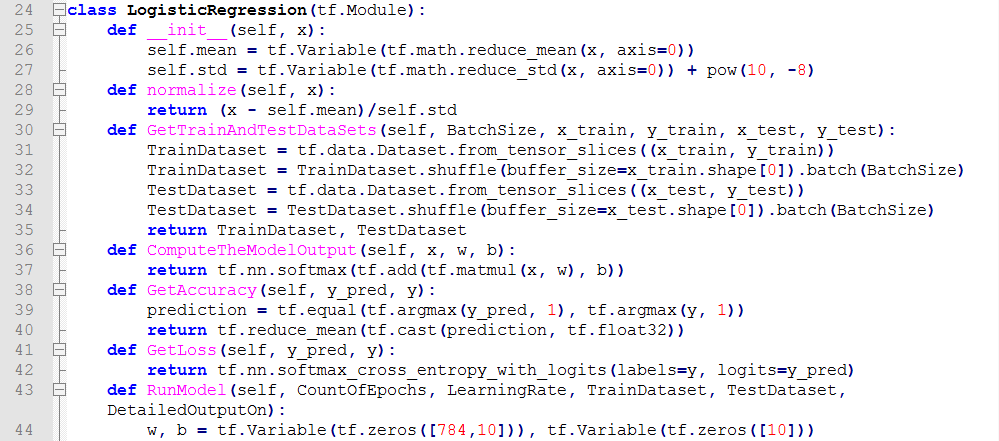
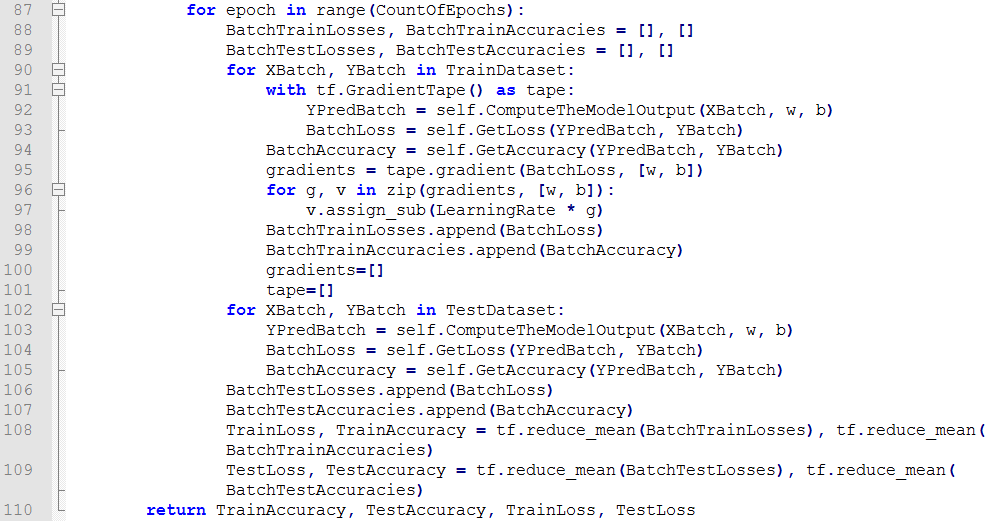
**Лабораторна робота 5 – Штучні нейронні мережі.**

**Завдання 1.** Розв'яжіть лабораторну №3 за допомогою нейронної мережі. Кількість шарів, нейронів, метод оптимізації – на Ваш розсуд. Дозволяється використання обгортки Keras для побудови моделі, однак реалізація на чистому TensorFlow (або PyTorch) заохочується додатковими балами.

Завдання було виконано за допомогою TensorFlow та Keras.

Код класу LogisticRegression з використанням чистого TensorFlow:





**Опис проведених досліджень**

Реалізація логістичної регресії за допомогою TensorFlow для вирішення задач багатокласової та бінарної класифікації дещо відрізняється: для обчислення функції активації та коефіцієнта втрат використовувався інструмент softmax замість sigmoid.

На відміну від попередньої роботи, ми аналізували як точність, так і значення втрат на тренувальних та тестових даних, а також кількість навчальних епох, проте вплив поліноміальності та коефіцієнтів alpha1 і alpha2 не досліджувались.

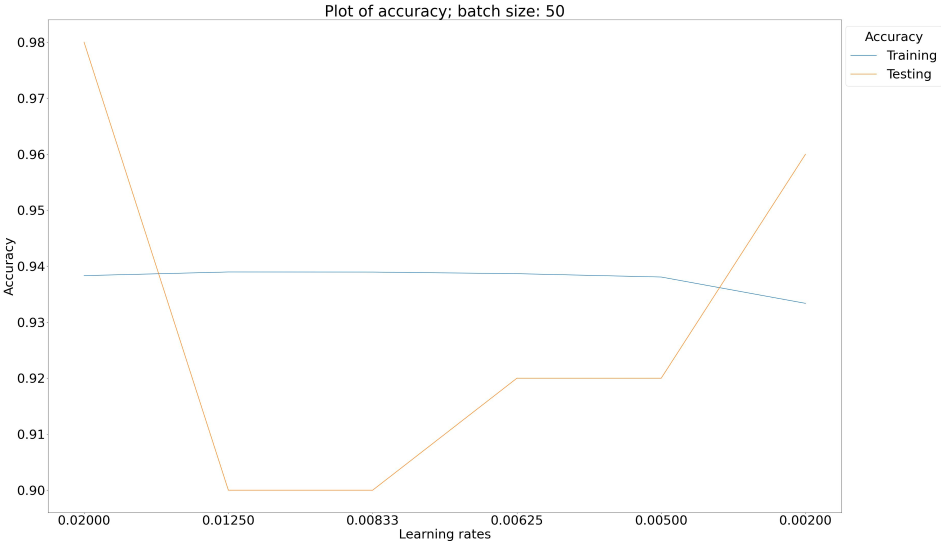
Запустивши код із стандартними параметрами, ми отримали наступний вивід:



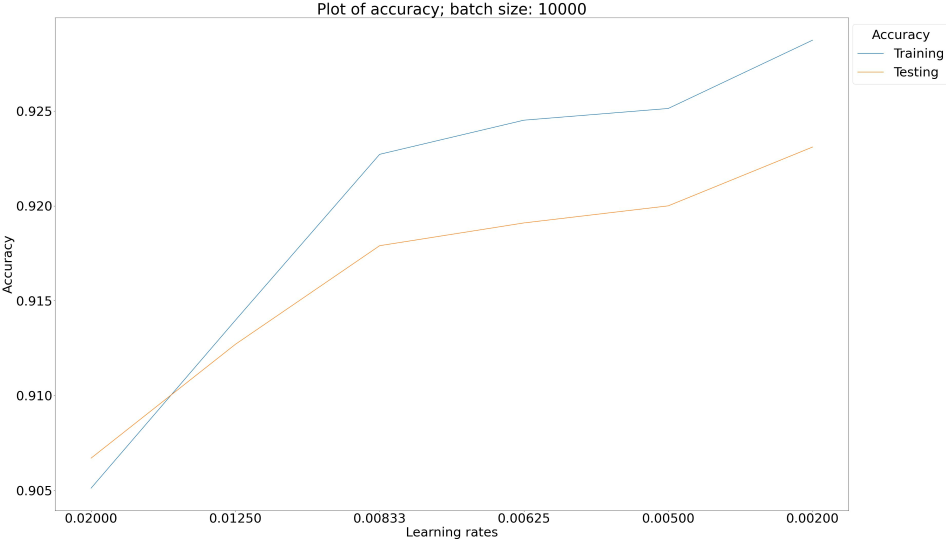
Видно, що точність нашої моделі на тестових даних, реалізованої за допомогою за допомогою TensorFlow, виявилась більшою на 6.56%.

Як і в попередніх роботах, з ціллю оптимізації ми винесли деякі підготовчі дії в окремий метод, який виконується перед початком нової серії експериментів з іншим розміром частини вибірки (BatchSize). Програма також містить код для визначення часових витрат, фіксації результатів у вигляді графіків, діаграм різних типів та текстового csv-файлу, який при необхідності може бути проаналізований табличним процесором.

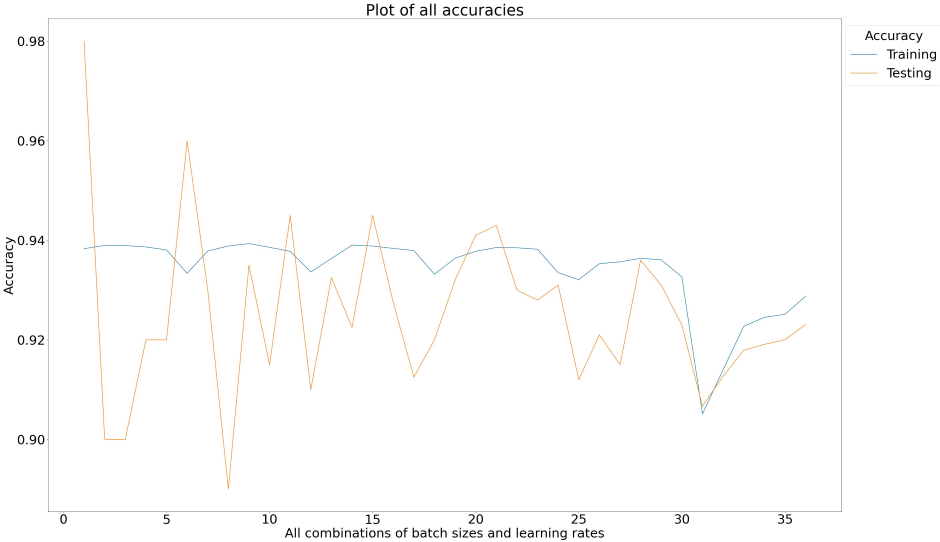
Тестування моделі включало в себе перебір різних розмірів частини вибірки (BatchSize) з коефіцієнтами швидкості навчання (LearningRate), а кількість епох при цьому була 10. Після визначення найкращих значень цих параметрів було виконано один експеримент із кількістю епох 1000. Наведемо графік, який ілюструє зміну точності при певному BatchSize з різними значеннями LearningRate:



Коефіцієнт швидкості навчання достатньо сильно впливає на якість моделі, проаналізованої на тестових даних, та менше впливає на точність при аналізі тренувальної вибірки. Схожу історію помітно при інших BatchSize окрім 10000:

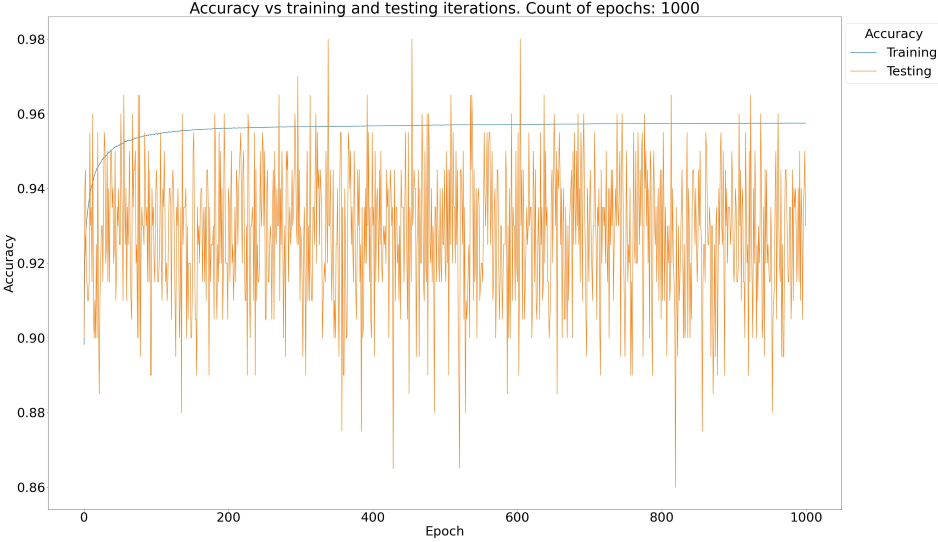


Тут ми бачимо правильну тенденцію, коли зростання коефіцієнта швидкості навчання призводить до збільшення точності моделі на всіх двох типах даних, причому рівень якості на тестовій вибірці у переважній більшості випадків нижчий за тренувальний. Графіки функції втрат є повністю оберненими до наданих. Проілюструємо глобальну зміну якості моделі при всіх комбінаціях описаних параметрів на тренувальних та тестових даних:



Видно, що точність, отримана на тренувальній вибірці, є більш стабільною.

Наведемо графік зміни якості моделі на кожній з епох:

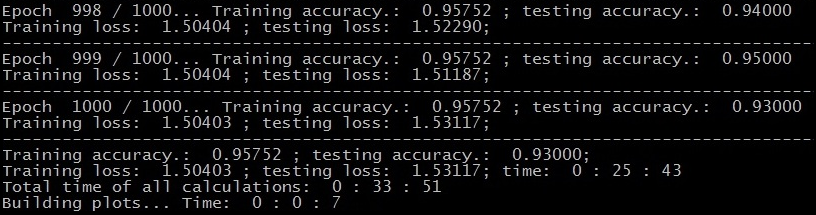


Ми бачимо подібну тенденцію: точність на тренувальній вибірці є набагато стабільнішою за точність на тестовій вибірці, і в більшості випадків вона є вищою.

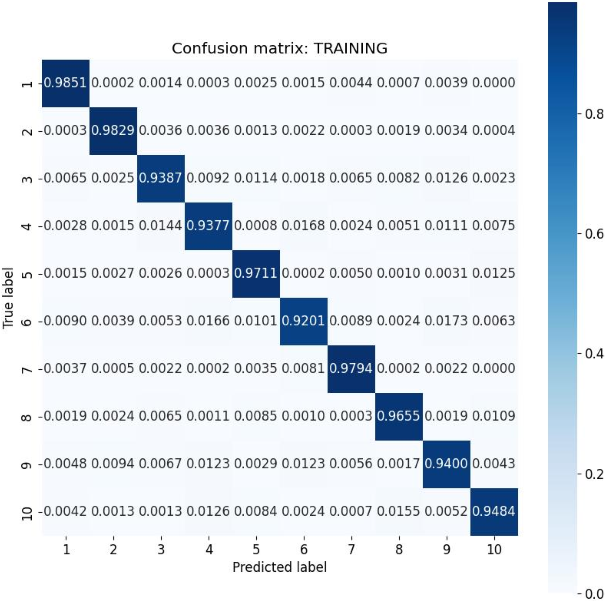
Такі ж результати показують і кольорові діаграми:

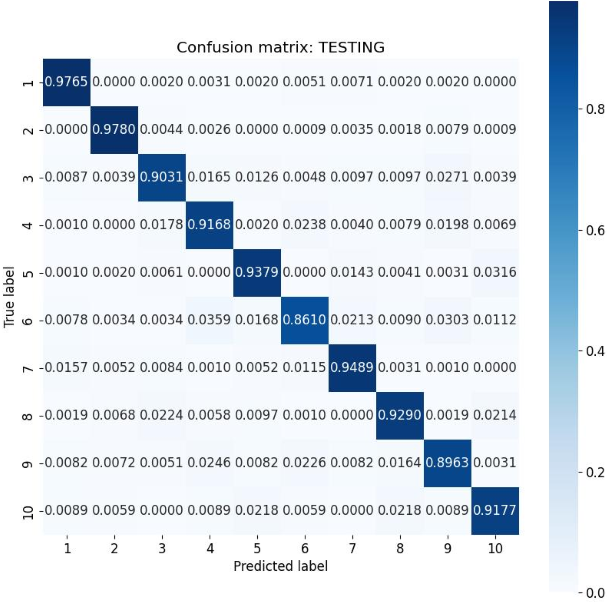
|  |  |
| --- | --- |
|  |  |

Після завершення всіх випробувань був отриманий наступний вивід на екран:

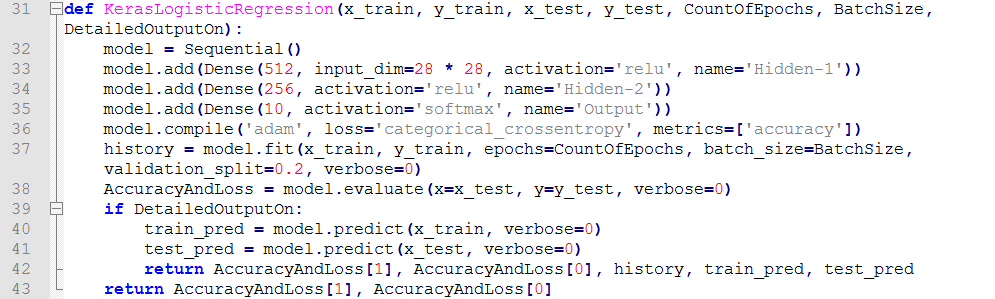


Для кращої моделі були побудовані діаграми типу «Confusion matrix»:





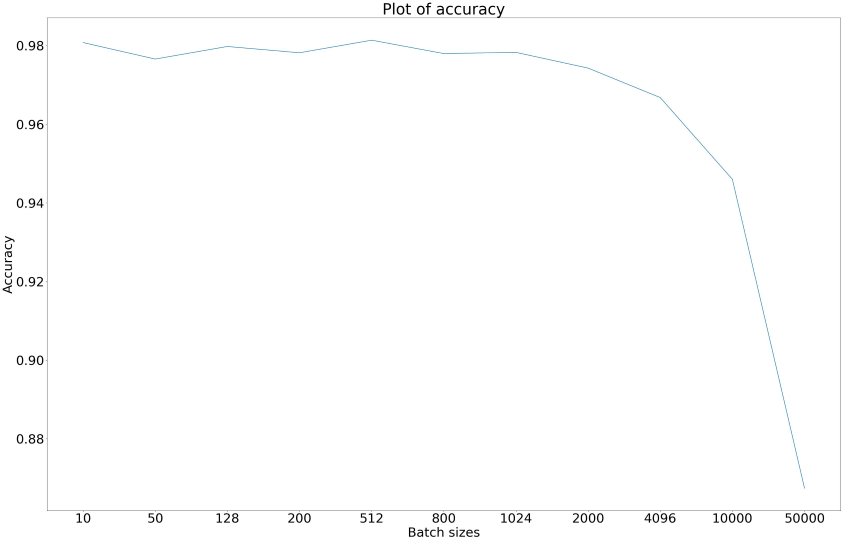
Завдання даної лабораторної роботи також було виконано за допомогою Keras:



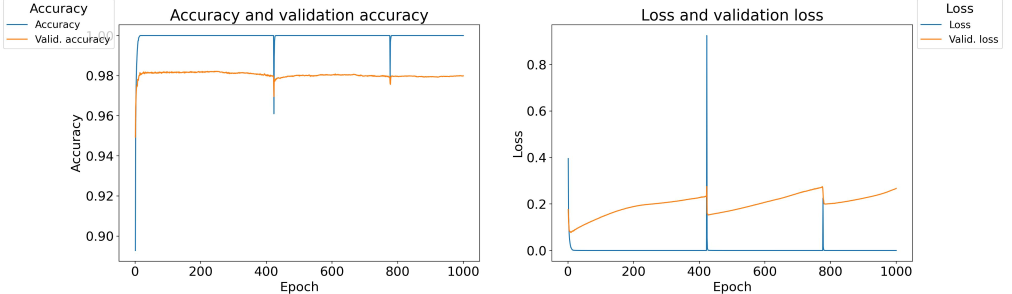
Ми бачимо, що модель, створена за допомогою Keras, майже на 5% точніша, ніж TensorFlow з параметрами, як у попередніх двох роботах:



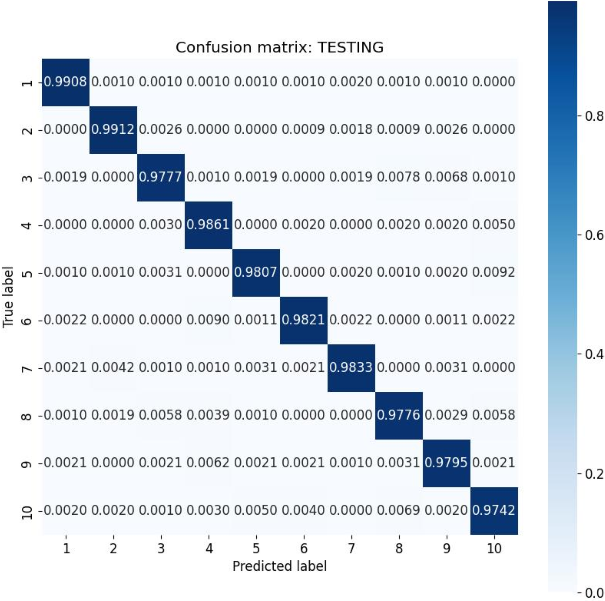
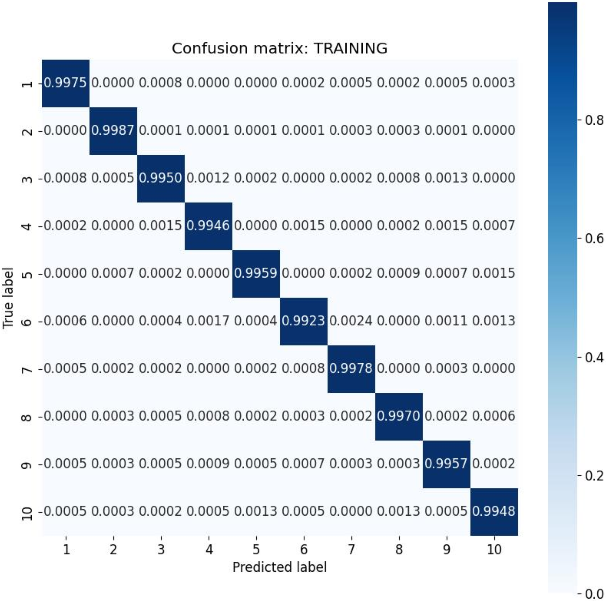
Оскільки обраний метод оптимізації («adam») автоматично задає значення параметра LearningRate (коефіцієнт швидкості навчання) в залежності від отриманого градієнта, ми його не аналізували, хоча в Keras, можливо, є способи змінити його. Тому ми перевіряли вплив BatchSize (розмір частини вибірки) на точність та коефіцієнт втрат лише на тестових даних, з кількістю епох 10:



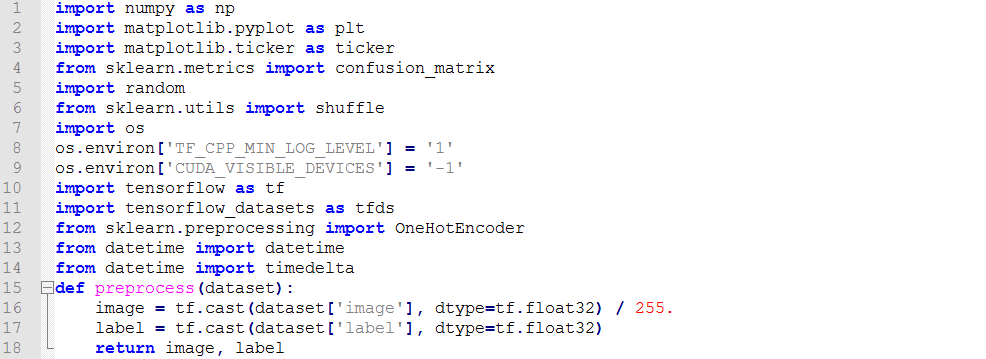
Видно, що розмір частини вибірки, більший за 2000, призводить до стрімкого погіршення якості моделі. Після отримання найкращого значення (512) було виконано один експеримент з кількістю епох 1000, отримано такі графіки:

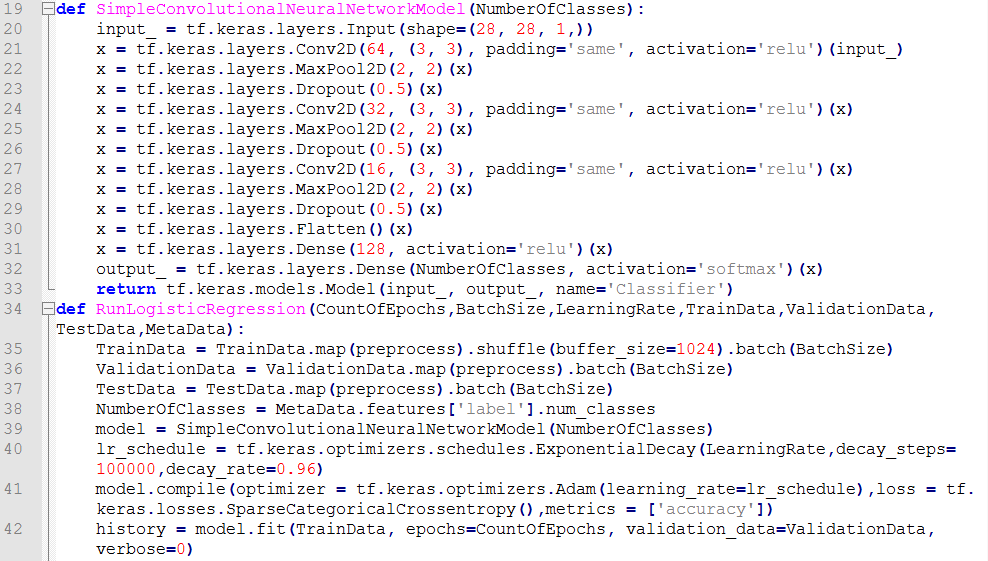


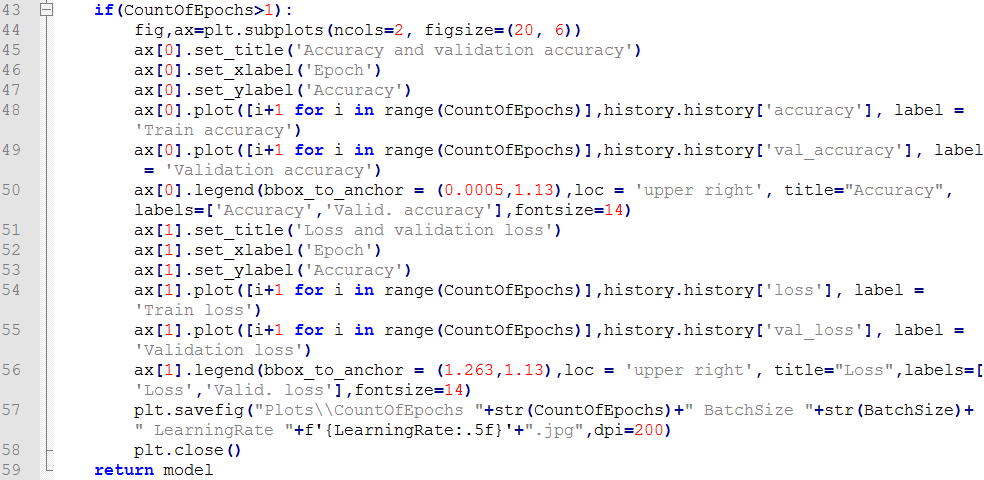
Можна помітити, що загальний рівень точності на тестових даних у переважній більшості випадків вищий, ніж на валідаційних.

Для кращої моделі були побудовані діаграми типу «Confusion matrix»:

Оскільки попереднє рішення має ряд недоліків, ми створили ще одне, з більшою кількістю шарів Keras, а також з можливістю аналізувати LearningRate:







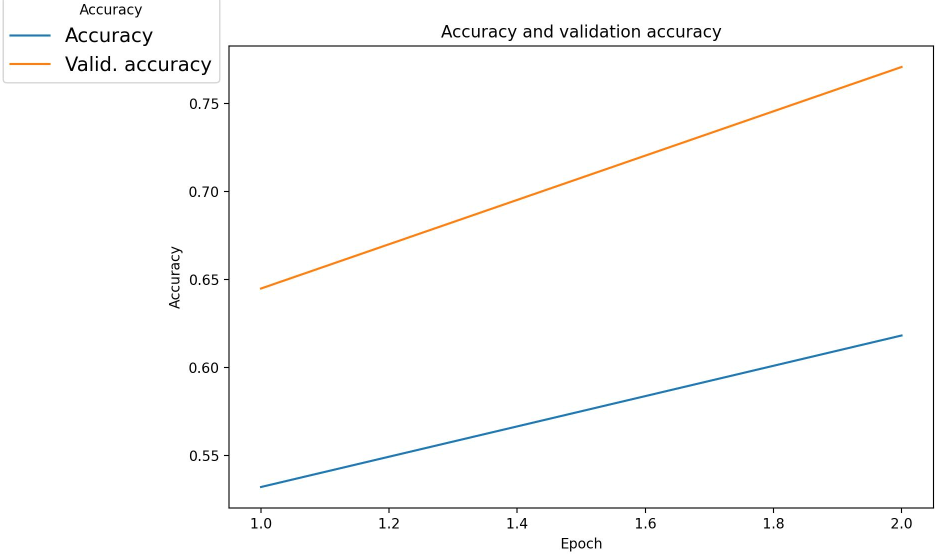
Запустивши код із стандартними параметрами, ми отримали наступний вивід:



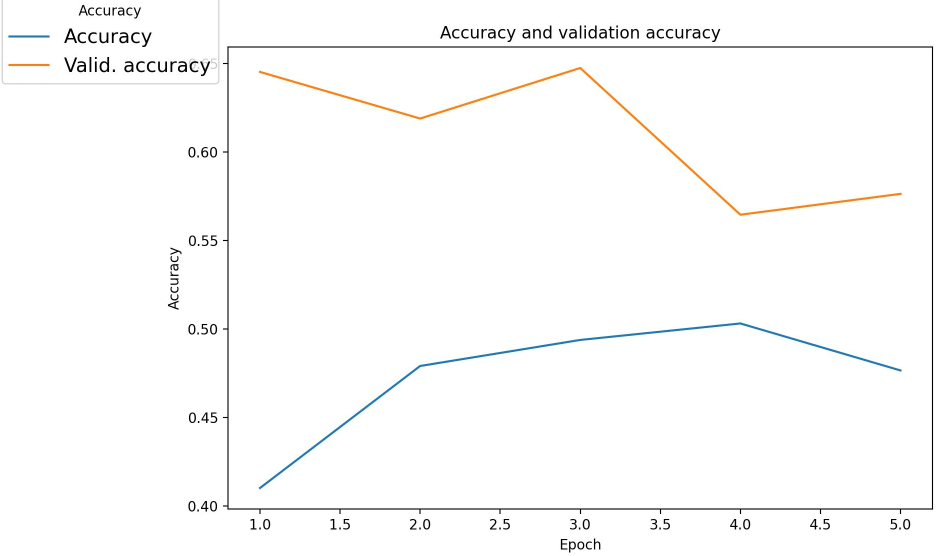
Бачимо, що точність даної моделі більша за попередню лише на майже 1%, проте обчислення тривали в рази довше з тими самими параметрами.

Нами було проведено багато експериментів з усіма комбінаціями кількості епох (від 1 до 5), розмірів частини вибірки (BatchSize) та коефіцієнтів швидкості навчання (LearningRate), для кожної з них було побудовано графіки такого вигляду:

CountOfEpochs 2 BatchSize 32 LearningRate 0.02000



CountOfEpochs 5 BatchSize 32 LearningRate 0.02000



Видно, що зі збільшенням кількості навчальних епох загальний рівень точності роботи моделі на валідаційних та тестових даних збільшується, а у переважній більшості випадків валідаційна точність вища за тестову.

Для кожної епохи також отримали більш загальні графіки та діаграми:

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

Підсумковий графік та кольорова діаграма виглядають таким чином:

