МІНІСТЕРСТВО ОСВІТИ І НАУКИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ.ІГОРЯ СІКОРСЬКОГО»

НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Лабораторна робота №3

«Прикладне програмне забезпечення навчання та тестування нейронної мережі для класифікації тестових даних. Аналіз впливу архітектури мережі та параметрів навчання на точність класифікації.»

Виконав:
Студент 3 курсу
Групи ФІ-21
Голуб Михайло

Перевірив:
Железняков. Д. О.

Київ 2025

ЗМІСТ

1. ЗАВДАННЯ ЛАБОРАТОРНОЇ РОБОТИ

1.1. Набір даних

- CIFAR-100 або інші складні набори даних для класифікації зображень
- Можна використовувати більш прості набори даних (MNIST, CIFAR-10), але оцінка не буде знижена
- Можна використовувати набори даних для більш складного завдання (додаткові бали)

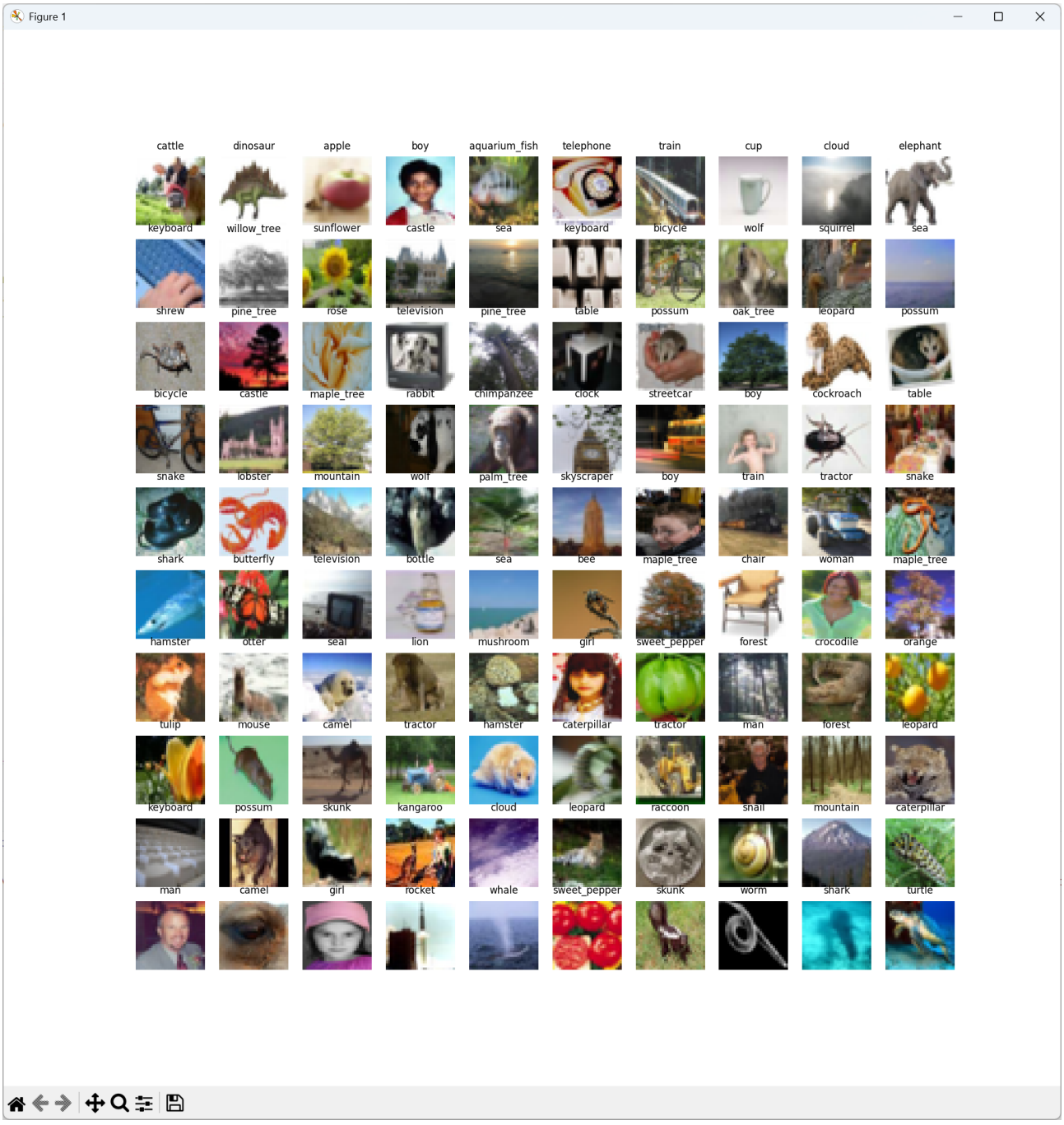
1.2. Завдання

- Ознайомитись з теоретичними відомостями до архітектур нейронних мереж (згорткові нейронні мережі, типи шарів, overfitting, ...).
- Завантажити набір даних
- За необхідності зробити попередню обробку та поділити датасет на: навчальний, валідаційний та тестовий
- Розробити архітектуру згорткової нейронної мережі для класифікації зображень
- Оцінити якість класифікації зображень (baseline)
- Провести серію експериментів (~ 4-8 експериментів) з архітектурою нейронної мережі та дослідити як впливає архітектура нейронної мережі на якість класифікації та процес навчання (loss, time). Бажано брати декілька різних параметрів. Для прикладу: збільшення кількості параметрів згорткового шару, інша функція активації, додавання Dropout, додавання нового скритого шару, тощо.
    - Зробити висновки по кожному експерименту.
- Порівняти результати. В деяких випадках доцільно показати у вигляді “Ablation Study”
    - Інколи бажано спробувати змінити один той самий параметр декілька разів (збільшити та зменшити)
    - Обрати найкращу модель та загальні висновки
- Зробити звіт
- Захистити роботу

2. ХІД РОБОТИ

2.1. Набір даних

Використовується набір CIFAR100, що містить 100 класів зображень. Сумарна кількість зображень – 60000, з них 50000 для тренування і 10000 для тестування. З 50000 тренувальних – 40000 для навчання, 10000 для передбачень і відбору.



(Мал. 1 Візуалізація класів)

2.2. Базова архітектура

2.2.1. Структура базової архітектури

Базова архітектура створюється наступним кодом:

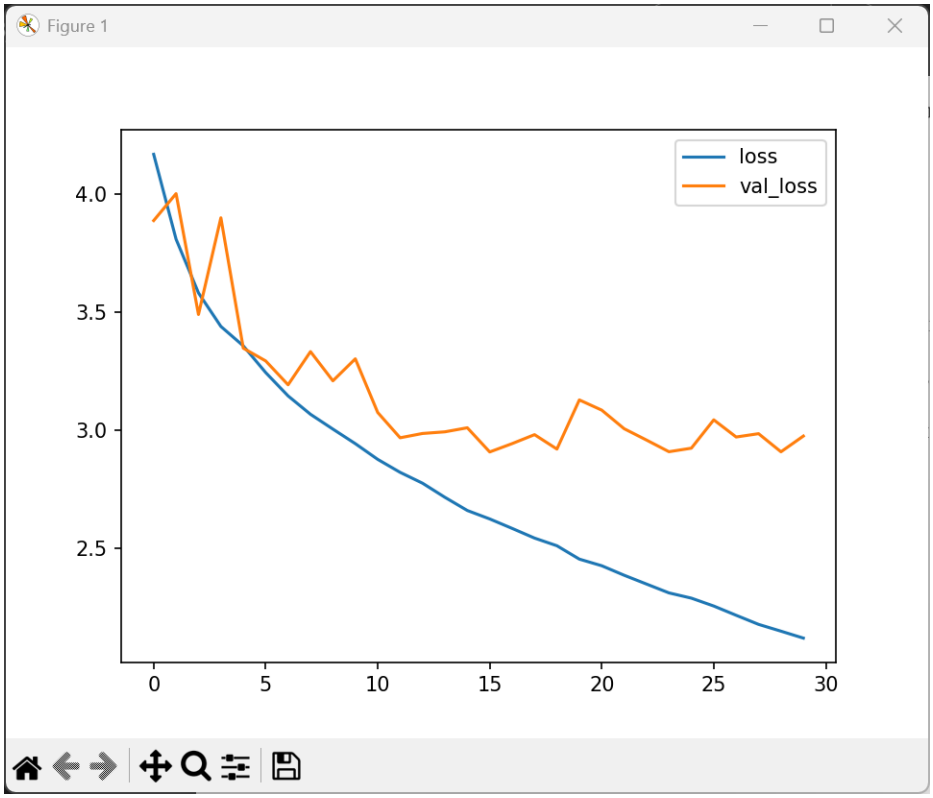
```
model = Sequential()
model.add(Input(shape=(32, 32, 3)))
model.add(Conv2D(32, (5, 5), activation='relu',padding='same'))
model.add(BatchNormalization())
model.add(Flatten())
model.add(Dense(400, activation='tanh'))
model.add(Dropout(0.3))
model.add(Dense(200, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(100, activation='softmax'))
model.compile(optimizer=Adam(learning_rate=0.001),
              loss=CategoricalCrossentropy(),
              metrics=['accuracy', TopKCategoricalAccuracy(k=2, name="Top2")])

history = model.fit(x_train, y_train_cat, batch_size=200, epochs = 30,
validation_data=(x_valid, y_valid_cat))
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 32)        2432

 batch_normalization (BatchN  (None, 32, 32, 32)       128
 ormalization)

 flatten (Flatten)           (None, 32768)             0

 dense (Dense)               (None, 400)               13107600

 dropout (Dropout)           (None, 400)               0

 dense_1 (Dense)             (None, 200)               80200

 dropout_1 (Dropout)         (None, 200)               0

 dense_2 (Dense)             (None, 100)               20100

=================================================================
Total params: 13,210,460
Trainable params: 13,210,396
Non-trainable params: 64
```

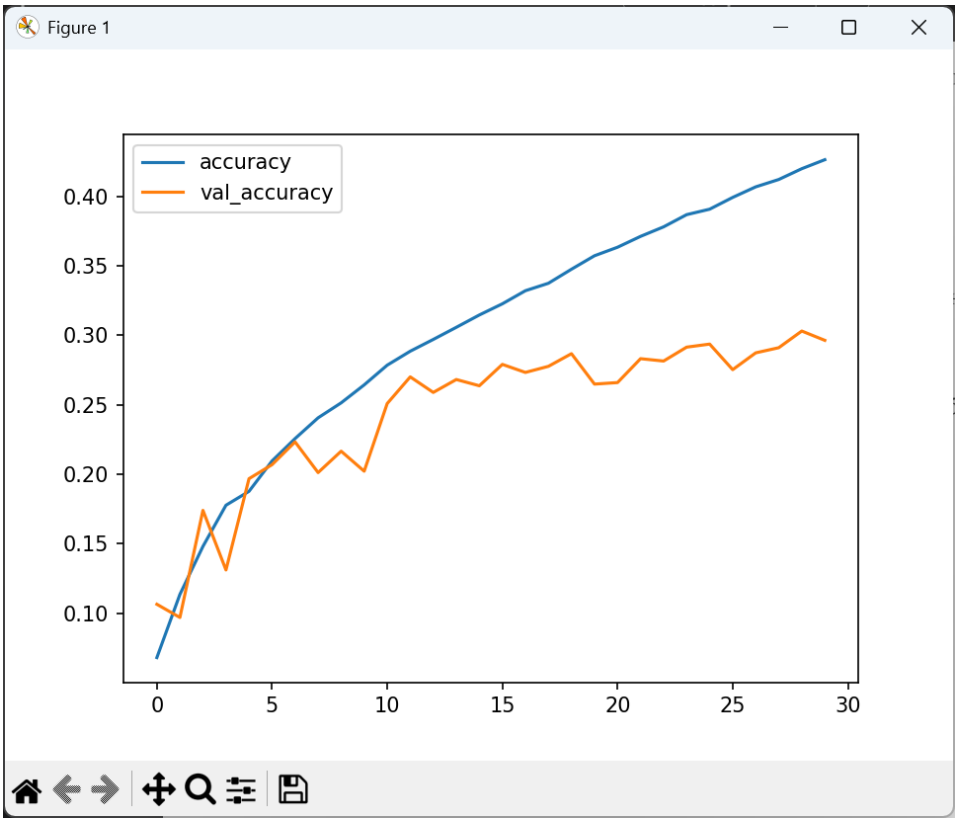
(Мал. 2 Базова архітектура. Результат виконання `model.summary()`)

2.2.2. Точність і втрати базової архітектури

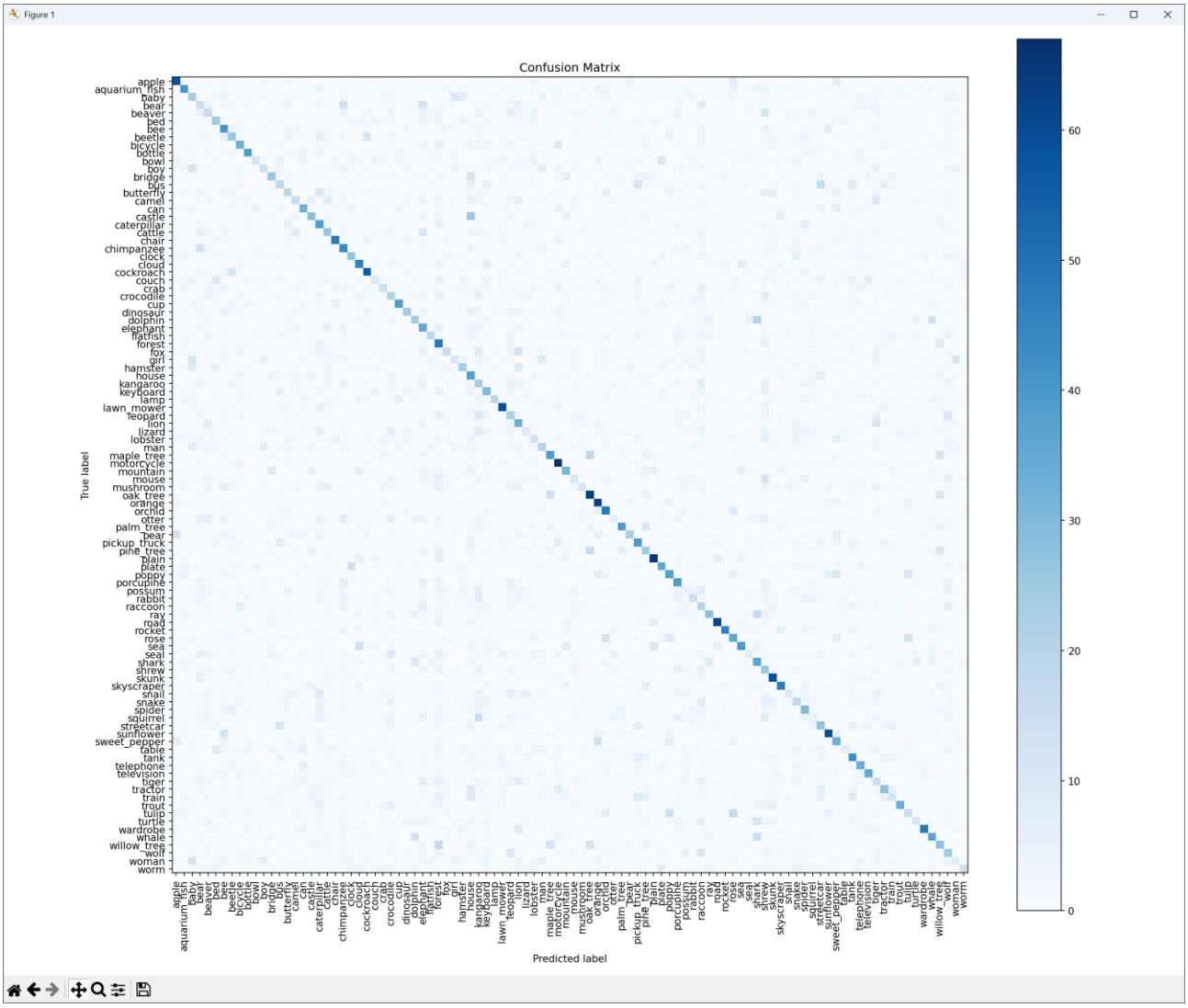
Базова архітектура перенавчається на 10-12 епосі, після чого модель дещо невпевнена і працює нормально (loss ~3, accuracy ~0.27)



(Мал. 3 Функції втрат базової архітектури)



(Мал. 4 Функції точності базової архітектури)



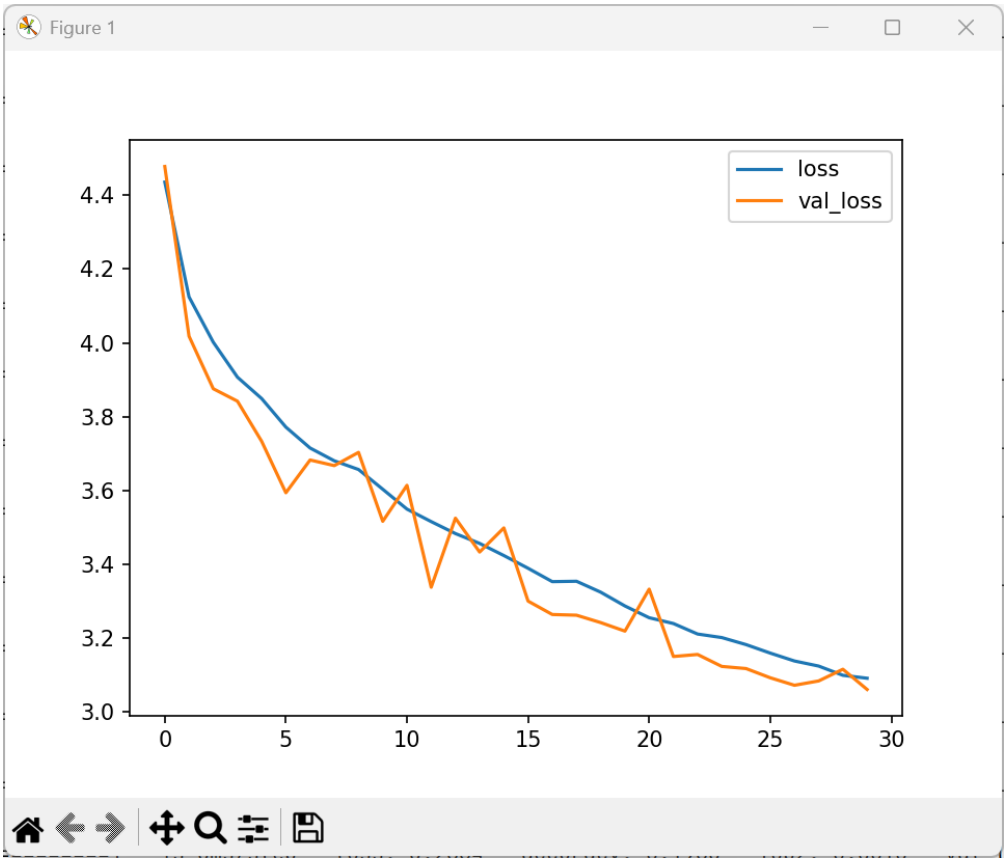
(Мал. 5 Матриця плутань базової архітектури)

З матриці плутань видно, що модель часто плутає схожі класи (дерева, дельфінів з акулами, тощо) і досить точно визначає унікальні класи (наприклад стільці).

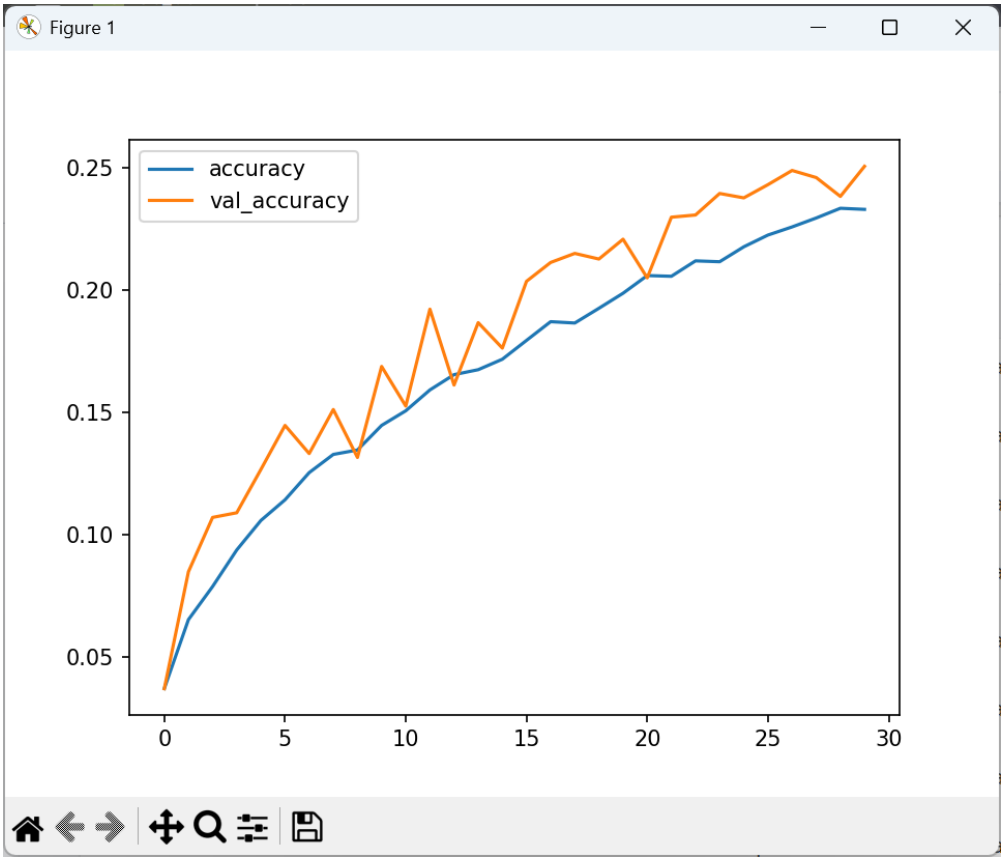
2.3. Експерименти

2.3.1. Значне збільшення Dropout

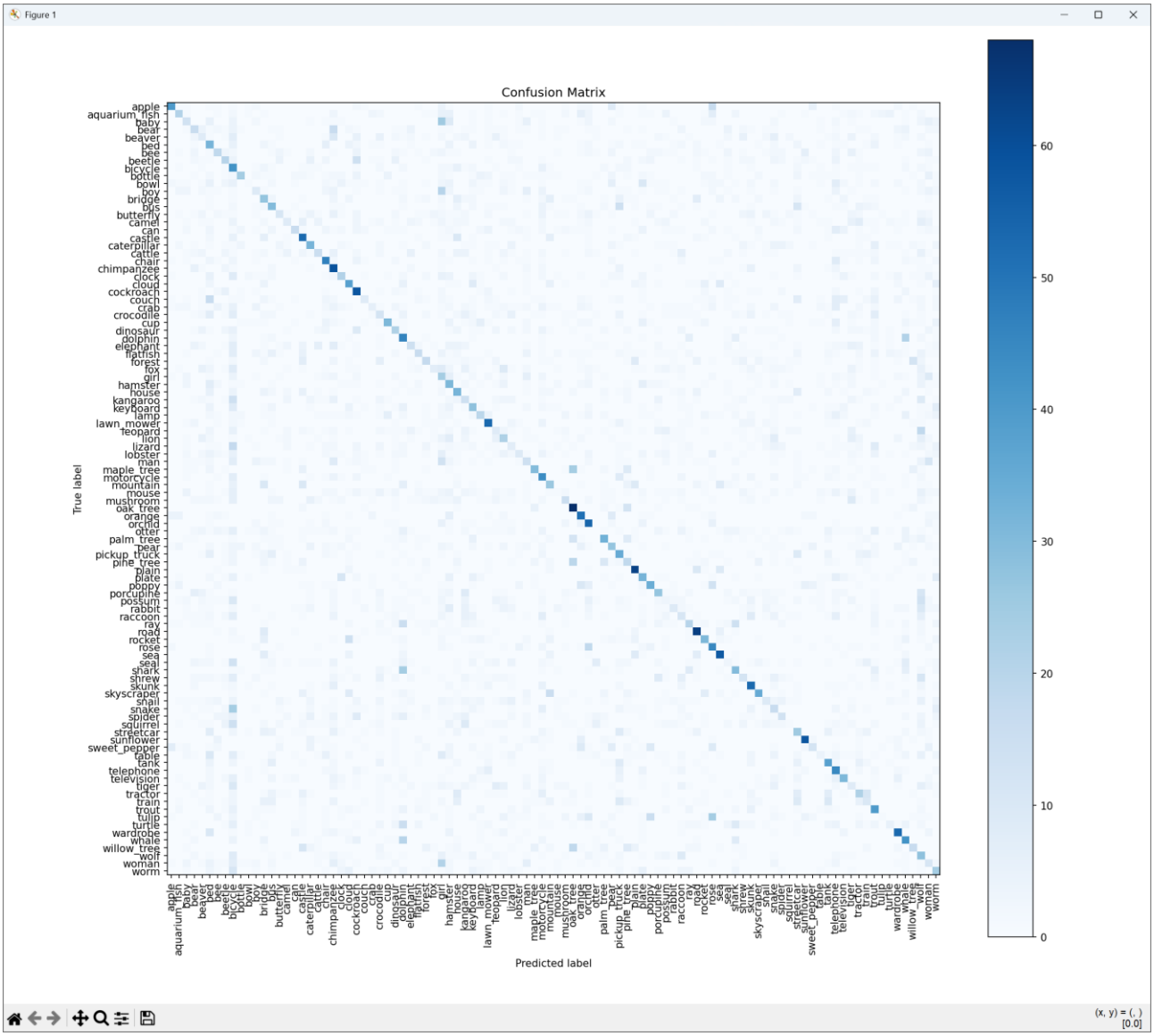
Модель перенавчається досить швидко, тож можна збільшити Dropout з 0.2 до 0.4:



(Мал. 6 Функції втрат для першого експерименту)



(Мал. 7 Функції точності для першого експерименту)

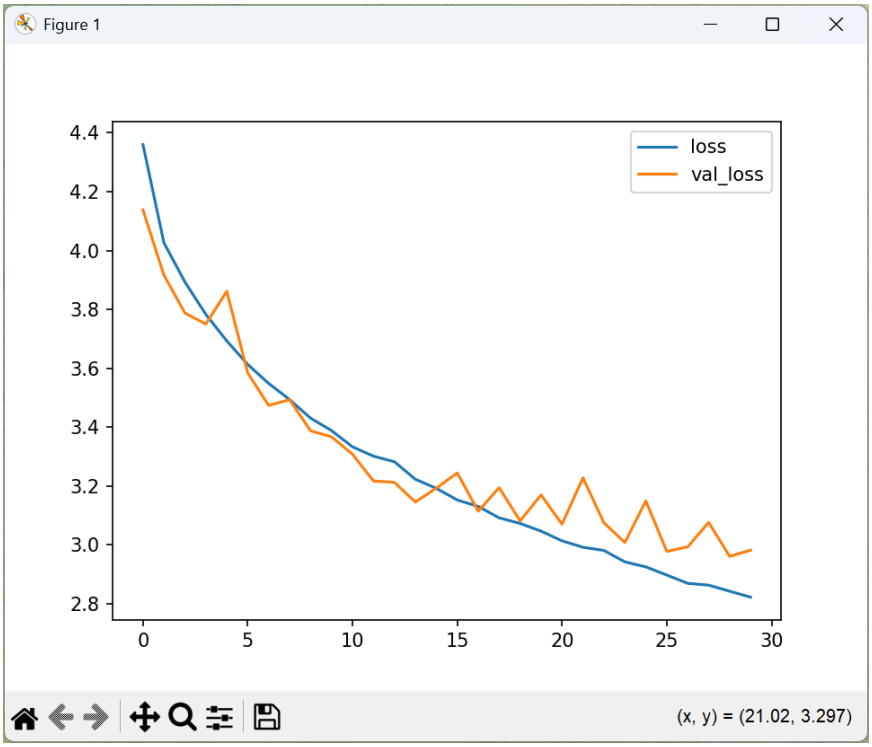


(Мал. 8 Матриця плутань для першого експерименту)

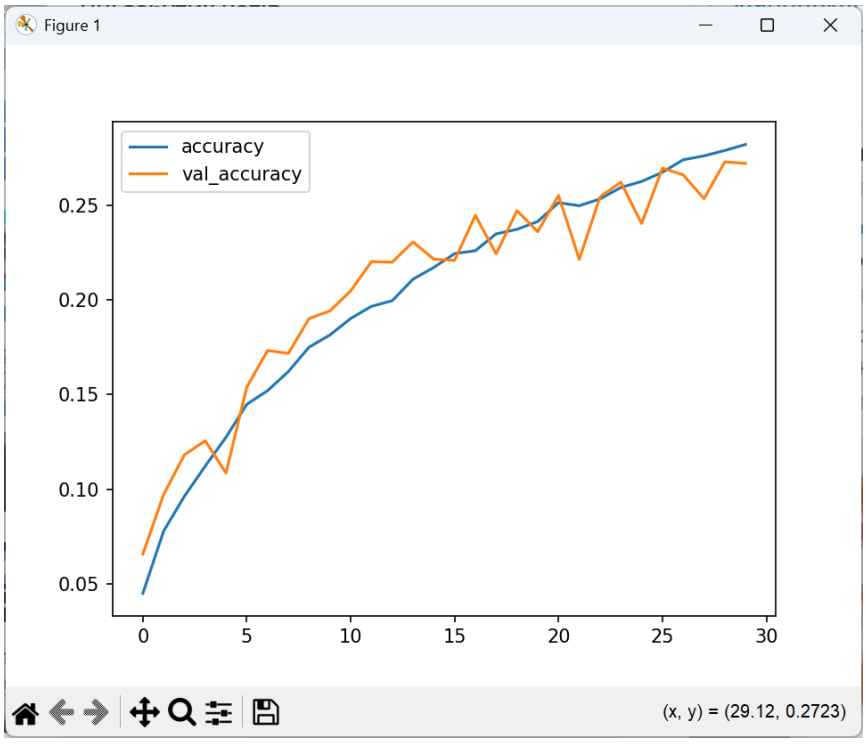
Отже, ця модель вже не перенавчається, але має схожу точність і гірші втрати.

2.3.2. Збільшення Dropout

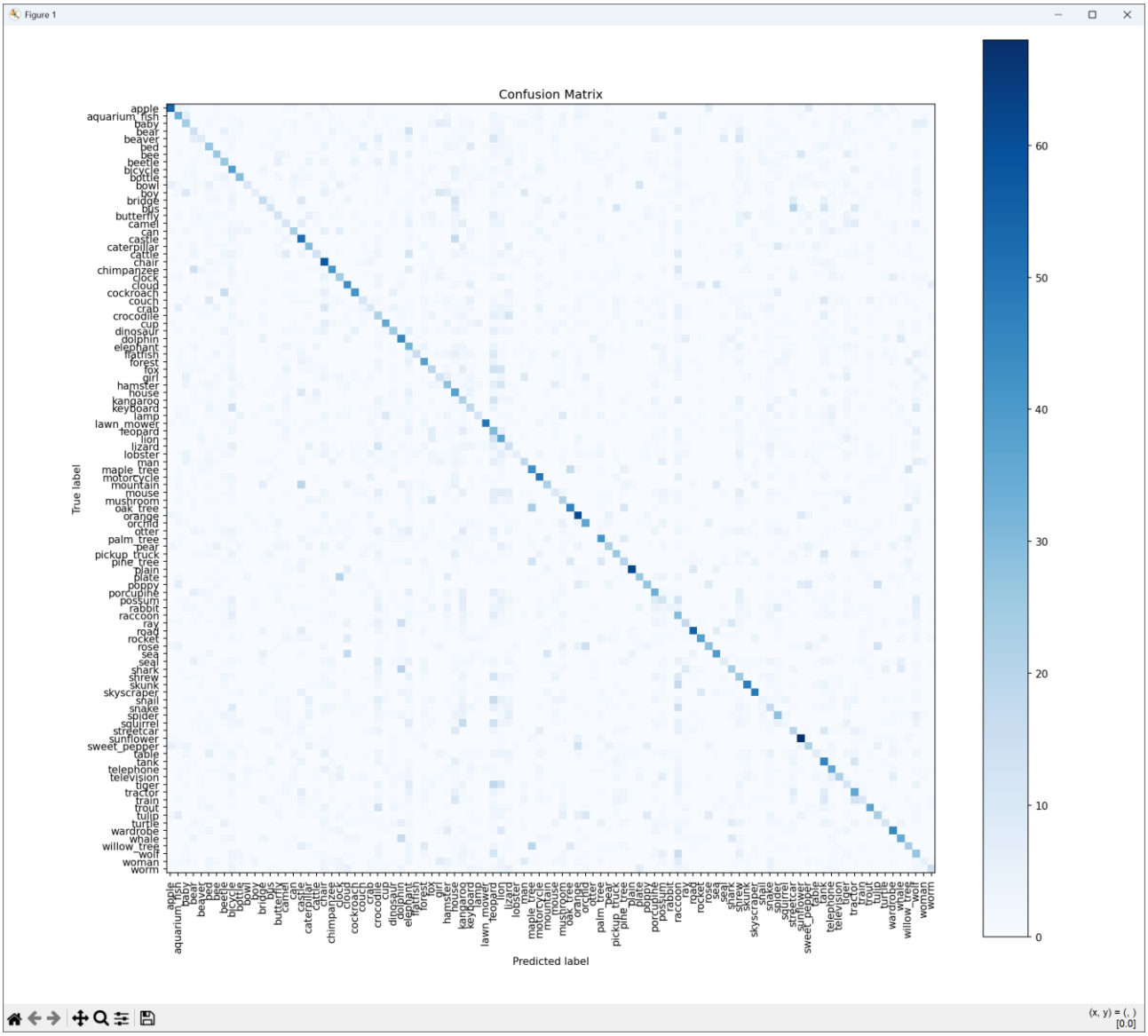
Встановлено значення dropout рівне 0.32:



(Мал. 9 Функція втрат для другого експерименту)



(Мал. 10 Функція точності для другого експерименту)

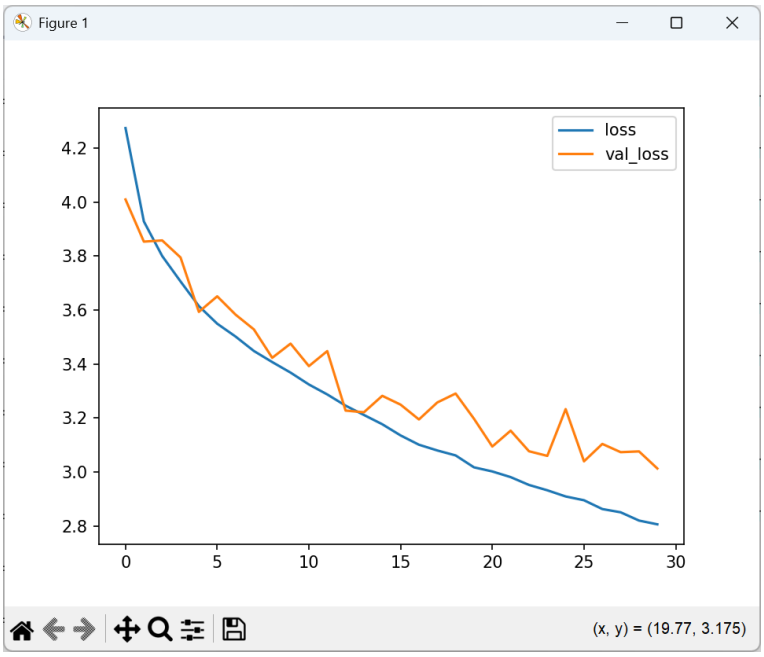


(Мал. 11 Матриця плутань для другого експерименту)

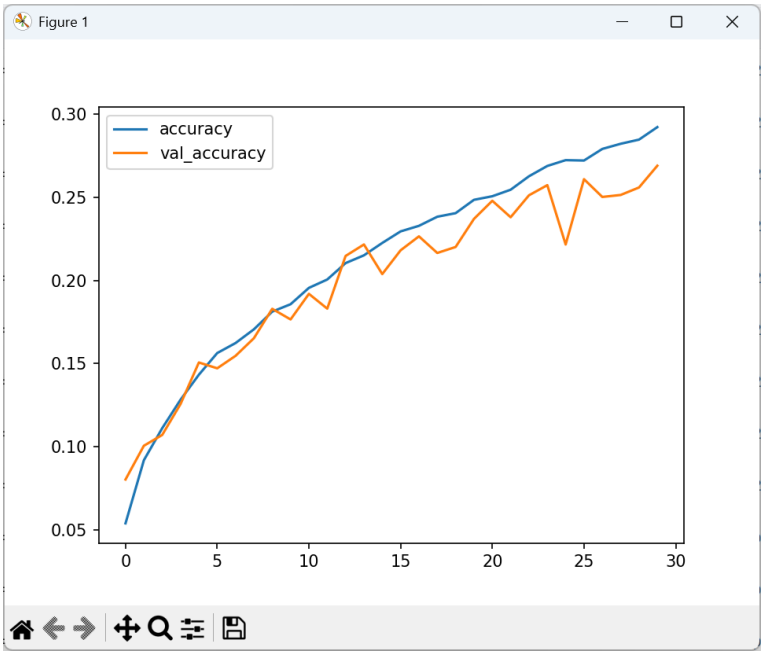
Дана модель має більшу точність і менші втрати ніж модель в першому експерименті, але меншу точність і схожі втрати ніж базова архітектура.

2.3.3. Збільшення розміру матриці Conv2D.

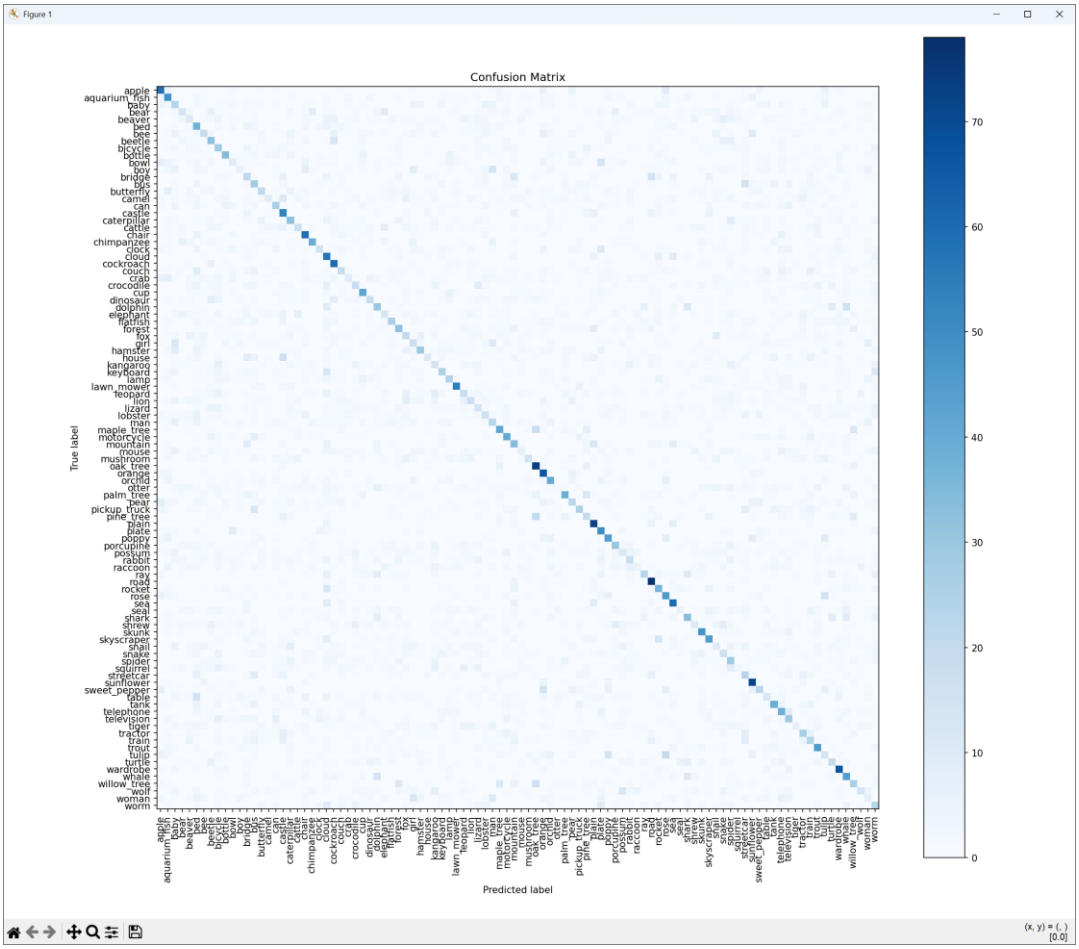
Відносно базової архітектури, збільшено розмір матриці в шарі Conv2D з 5x5 на 9x9. Dropout рівний 0.2:



(Мал. 12 Функція втрат для третього експерименту)



(Мал. 13 Функція точності для третього експерименту)

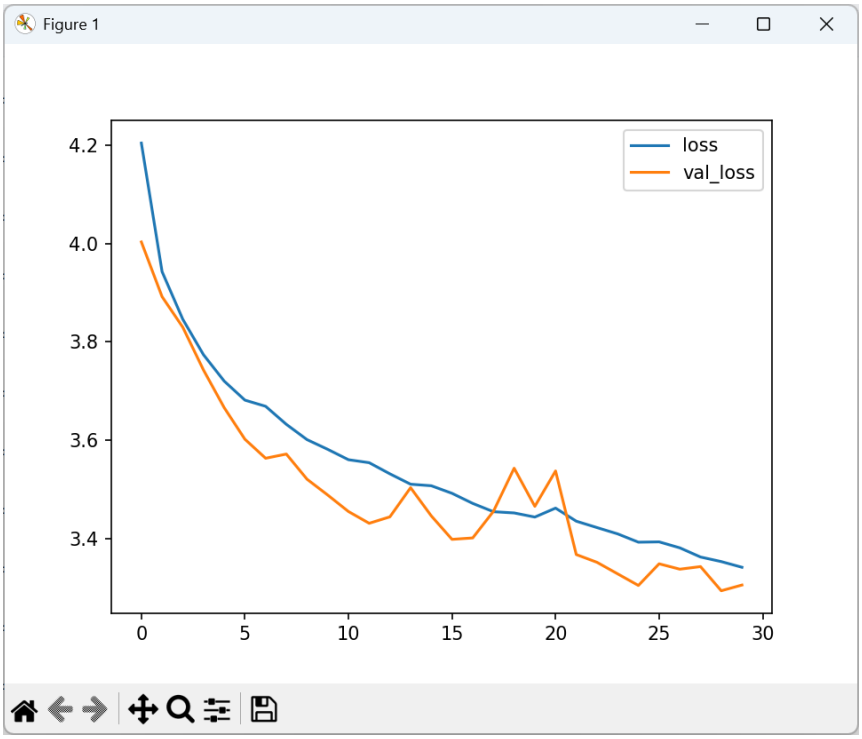


(Мал. 14 Матриця плутань для третього експеременту)

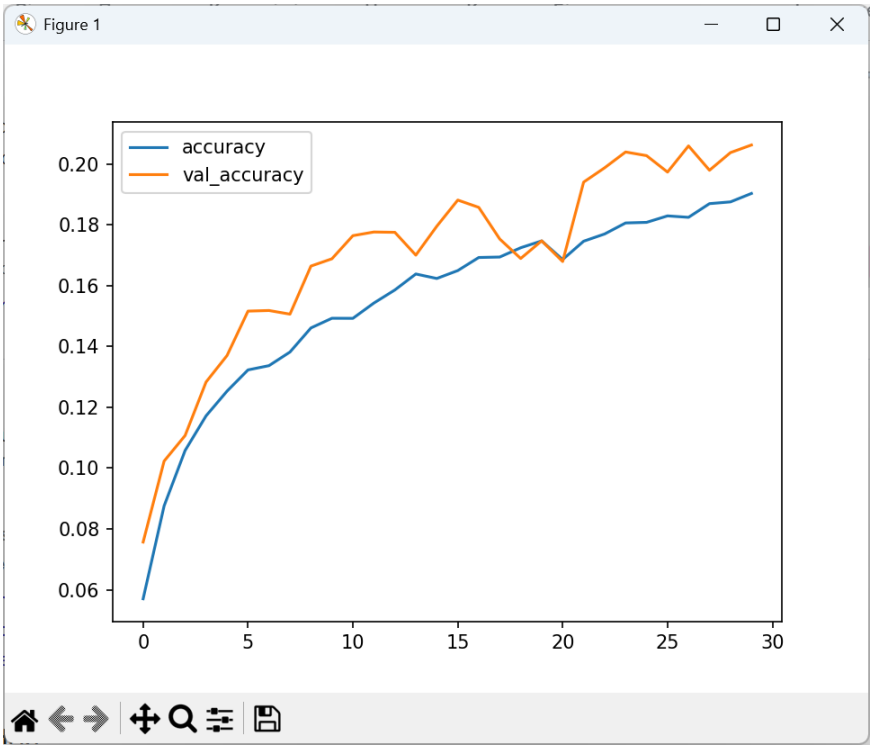
Точність цієї моделі співставна з усіма іншими, але ця модель має більше втрат.

2.3.4. Значне зменшення batch_size

Зменшено `batch_size` з 200 до 32:



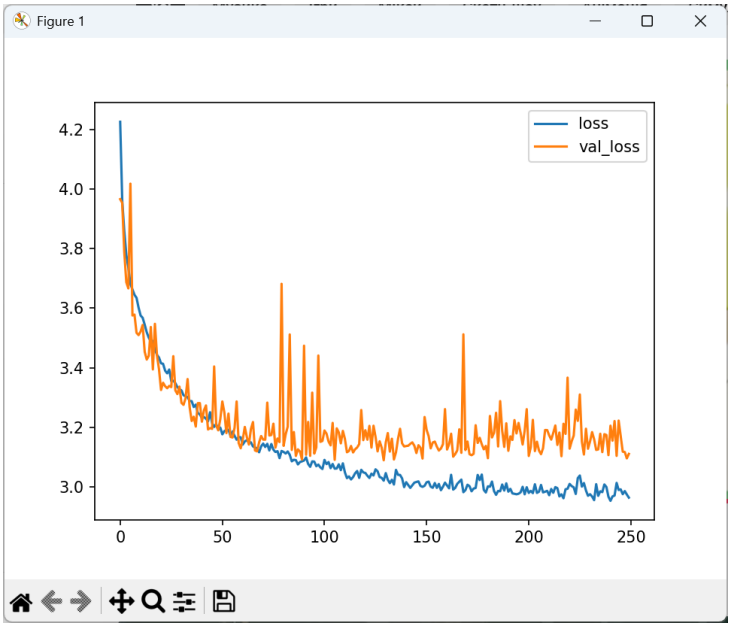
(Мал. 15 Функція втрат для четвертого експерименту)



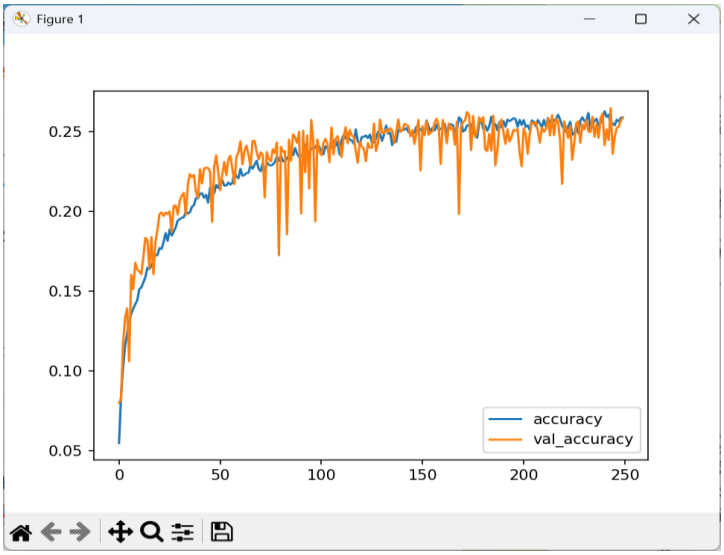
(Мал. 16 Функція точності для четвертого експерименту)

Модель недонавчена, кількість епох збільшено.

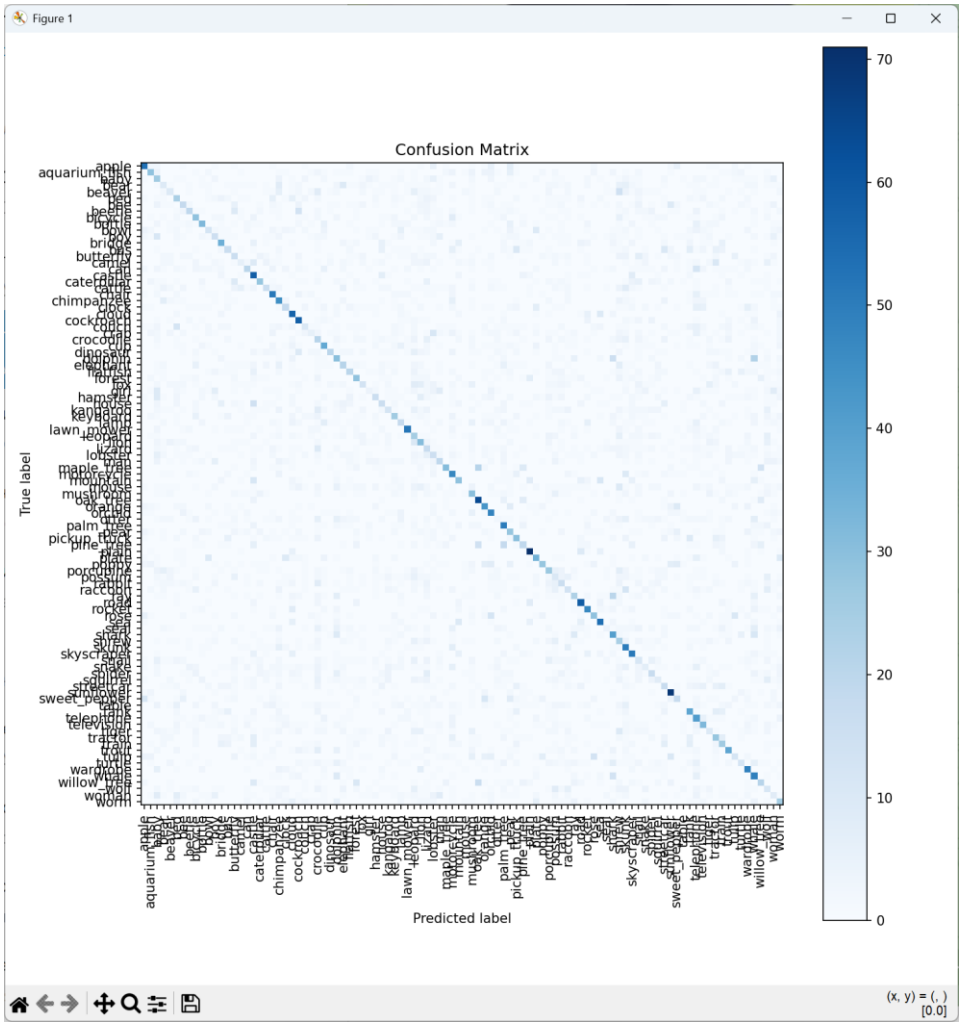
В тестах з 60, 100 і 250 епох модель досягає локального мінімуму. Результати навчання моделі протягом 250 епох:



(Мал. 17 Функція втрат для модифікованого четвертого експерименту)



(Мал. 18 Функція точності для модифікованого четвертого експерименту)



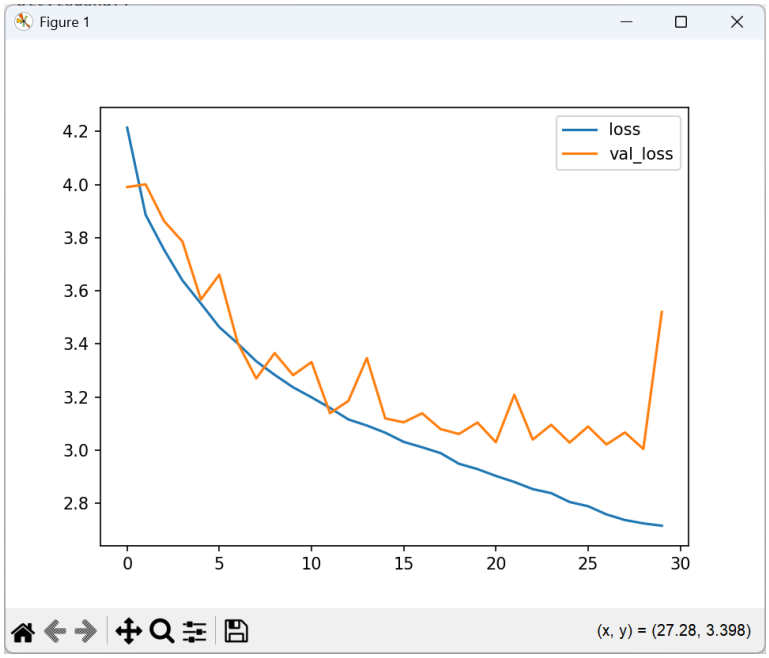
(Мал. 19 Матриця плутань для модифікованого четвертого експерименту)

Отже, зменшення batch_size лише сповільнює навчання моделі, не покращуючи її.

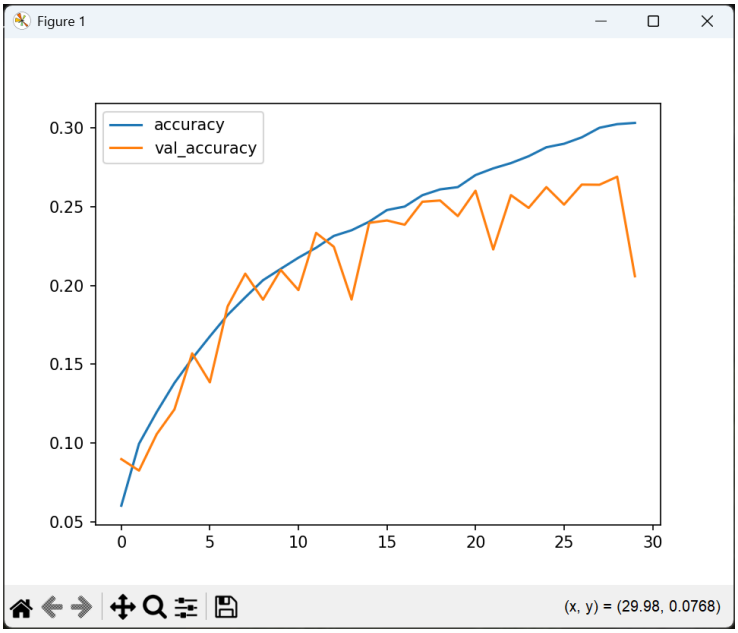
Оскільки модель потрапила в локальний мінімум, значення `batch_size` має бути більшим за 32.

2.3.5. Зменшення batch_size

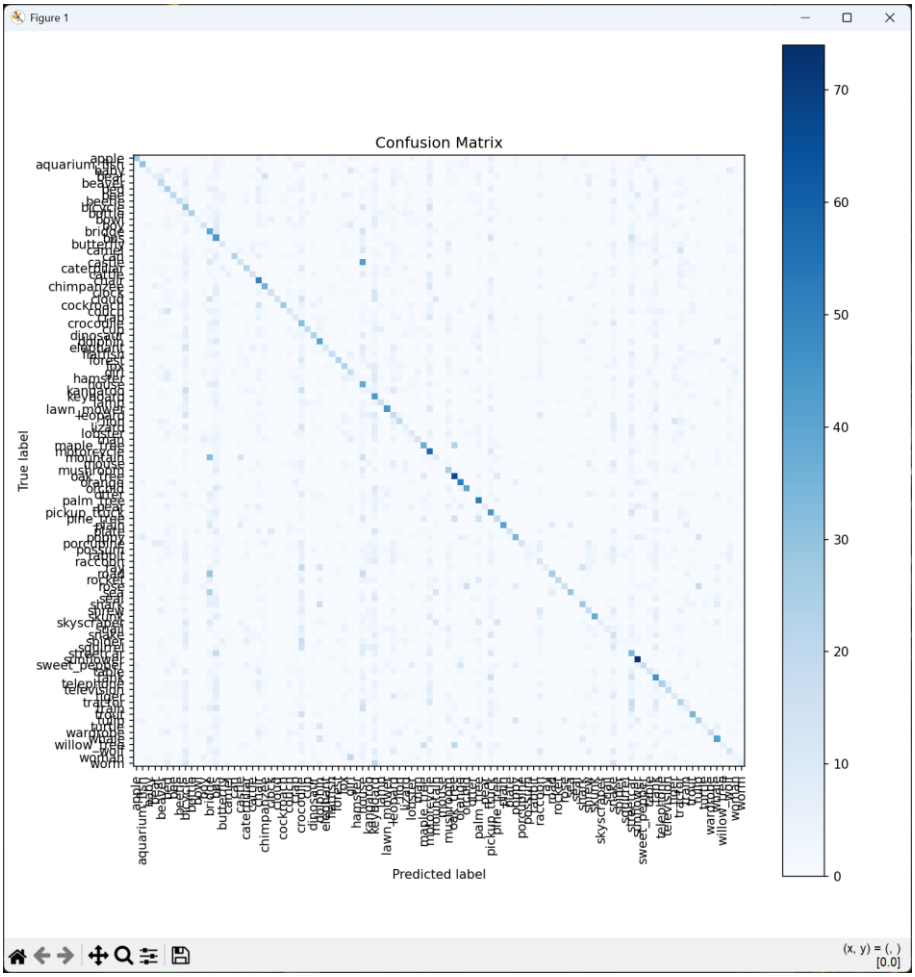
Покладено значення `batch_size` рівне 100:



(Мал. 20 Функція втрат для п’ятого експерименту)



(Мал. 21 Функція точності для п’ятого експерименту

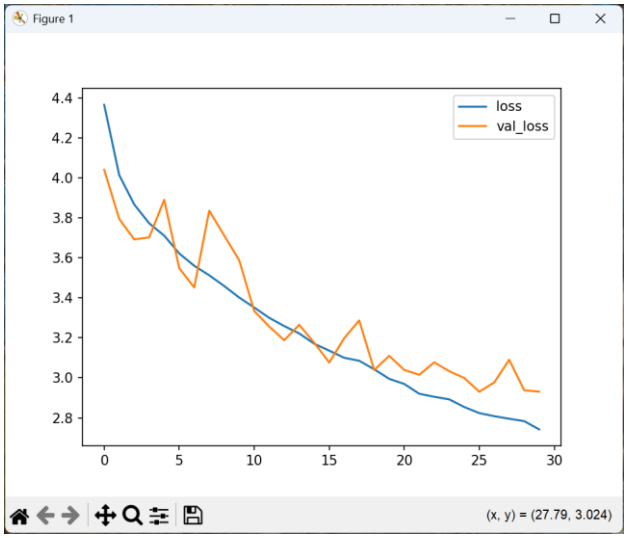


(Мал. 22 Матриця плутань для п’ятого експерименту)

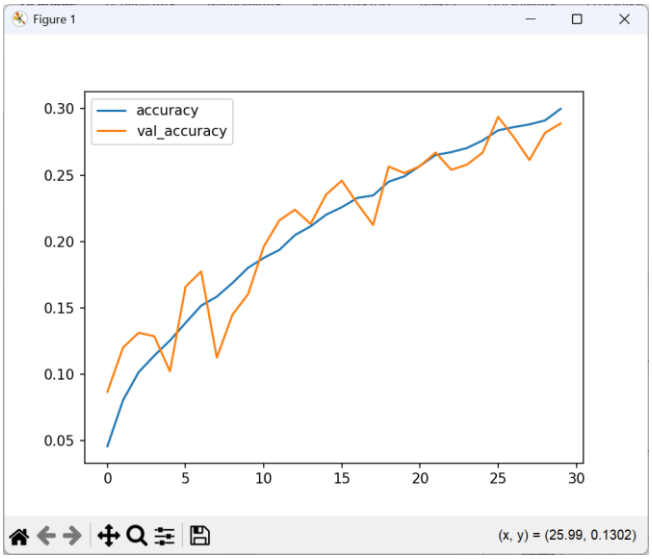
Дана модель на 29ій епосі була найбільш точною (0.265) з моделей усіх попередніх експериментів.

2.3.6. Подвоєння кількості нейронів в суцільних шарах

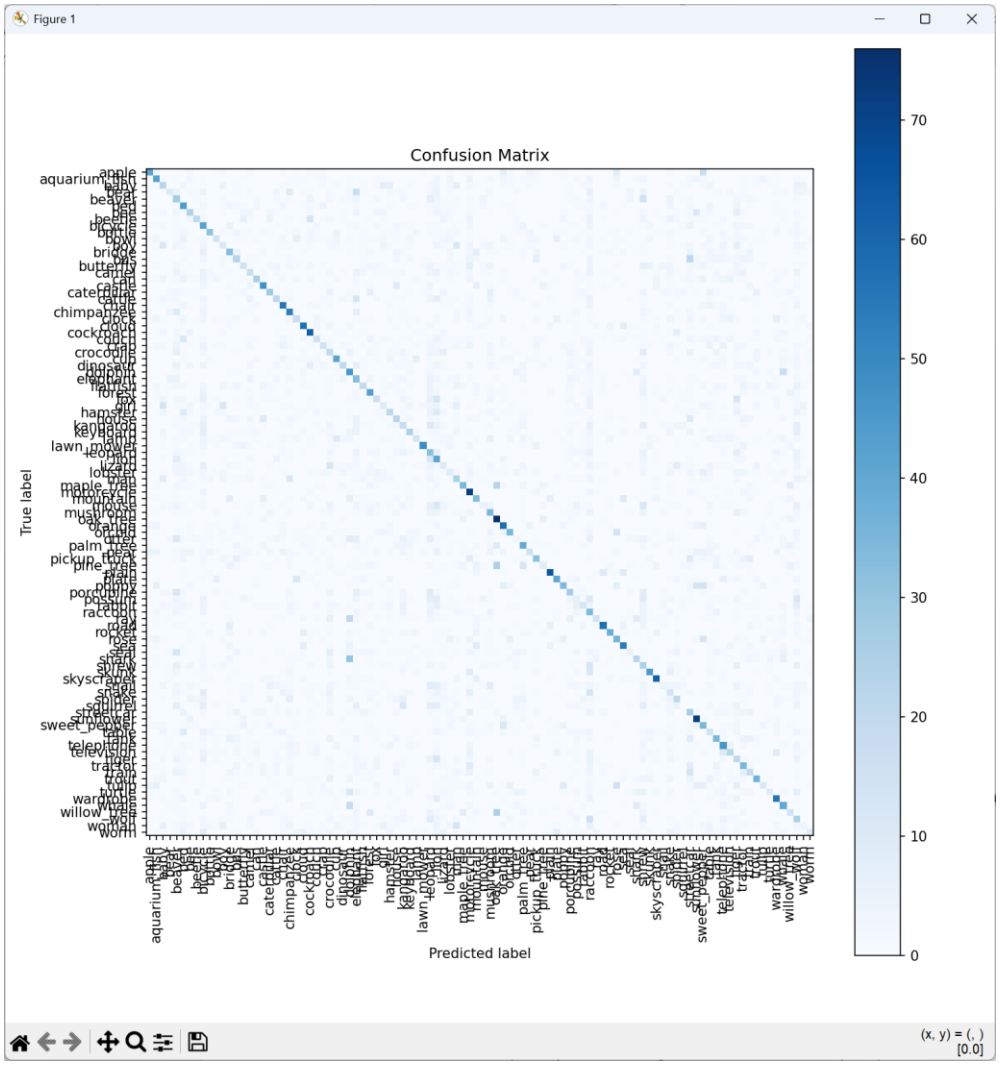
Подвоєно кількість нейронів в суцільних прихованих шарах. Для запобігання перенавчанню Dropout збільшено до 0.4



(Мал. 23 Функція втрат для шостого експерименту)



(Мал. 24 Функція точності для шостого експерименту)



(Мал. 25 Матриця плутань для шостого експерименту)

Збільшення кількості нейронів призвело до збільшення точності на 1% (відносно базової архітектури), до 27%

3. ВИСНОВКИ

- Усі моделі мають схожу точність 0.25-0.27 і схожі втрати 2.8-3.2. Оскільки точність простих моделей на CIFAR100 очікується 0.3-0.4, то можна вважати модель гіршою за стандартні.
- Обрана базова архітектура моделі є поганою і зміни одного-двох параметрів не призводять до значних змін. Скоріше за все це викликано використанням tanh.
- Найкращою з усіх моделей виявилась модель з подвоєною кількістю нейронів.
- Значне зменшення batch_size значно збільшує час тренування моделі, не покращуючи її характеристики.