Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

Кафедра прикладной математики и кибернетики

Отчёт по лабораторной работе №4

«CIFAR10 small images classification dataset»

Выполнил:

студент группы ИП-113

Шпилев Д. И.

ФИО студента

Работу проверил: Дементьева К. И.

ФИО преподавателя

Новосибирск 2024 г.

# **Задание**

Целью данной лабораторной работы является разработка нейронной

сети для решения задачи классификации или регрессии в зависимости от

набора данных в рамках варианта. Лабораторная работа предполагает

разработку на языке программирования Python с использованием библиотеки

Keras.

Распознавание 10 предметов на уменьшенных изображениях

(CIFAR10 small images classification dataset).

Все наборы данных доступны по ссылке: https://keras.io/api/datasets/

При разработке нейронной сети следует соблюсти наличие необходимых

составляющих исходя из следующих вариантов:

Нейросеть должна состоять из трёх полносвязных слоёв,

обязательное использование Dropout, в качестве оптимизатора

использовать Adam.

Выбор количества нейронов на всех внутренних слоях, функций

активации и других параметров должен быть обусловлен оптимальностью

работы модели.

# **Код программы**

from tensorflow.keras.datasets import cifar10

from tensorflow.keras import layers, models

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.utils import to\_categorical

(x\_train, y\_train), (x\_test, y\_test) = cifar10.load\_data()

x\_train = x\_train.astype('float32') / 255.0

x\_test = x\_test.astype('float32') / 255.0

y\_train = to\_categorical(y\_train, 10)

y\_test = to\_categorical(y\_test, 10)

model = models.Sequential()

model.add(layers.Flatten(input\_shape=(32, 32, 3)))

model.add(layers.Dense(128, activation='relu'))

model.add(layers.Dropout(0.2))

model.add(layers.Dense(64, activation='relu'))

model.add(layers.Dropout(0.2))

model.add(layers.Dense(10, activation='softmax'))

model.compile(optimizer=Adam(), loss='categorical\_crossentropy', metrics=['accuracy'])

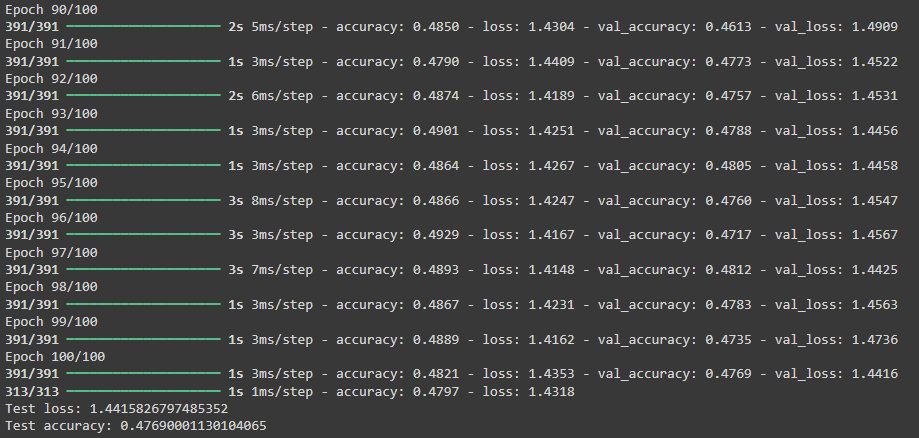
history = model.fit(x\_train, y\_train, epochs=20, batch\_size=128, verbose=1, validation\_data=(x\_test, y\_test))

test\_loss, test\_acc = model.evaluate(x\_test, y\_test)

print(f"Test loss: {test\_loss}")

print(f"Test accuracy: {test\_acc}")

# **Результаты работы**



# **Выводы**

После обучения 100 эпох получен результат – 0.477