МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА - Российский технологический университет»
РТУ МИРЭА

Институт Информационных Технологий
Кафедра Вычислительной Техники (ВТ)

Отчёт

по дисциплине

«Архитектура устройств и систем вычислительной техники»

Выполнили студенты группы ИВМО-02-24 Кутепов А.О.
Рьянов А.Е.
Смирнов А.В.

Принял преподаватель

Гуличева А.А.

Работа выполнена «__»_____20___г

«Зачтено»
«___»_____20___г

Москва 2024

Оглавление

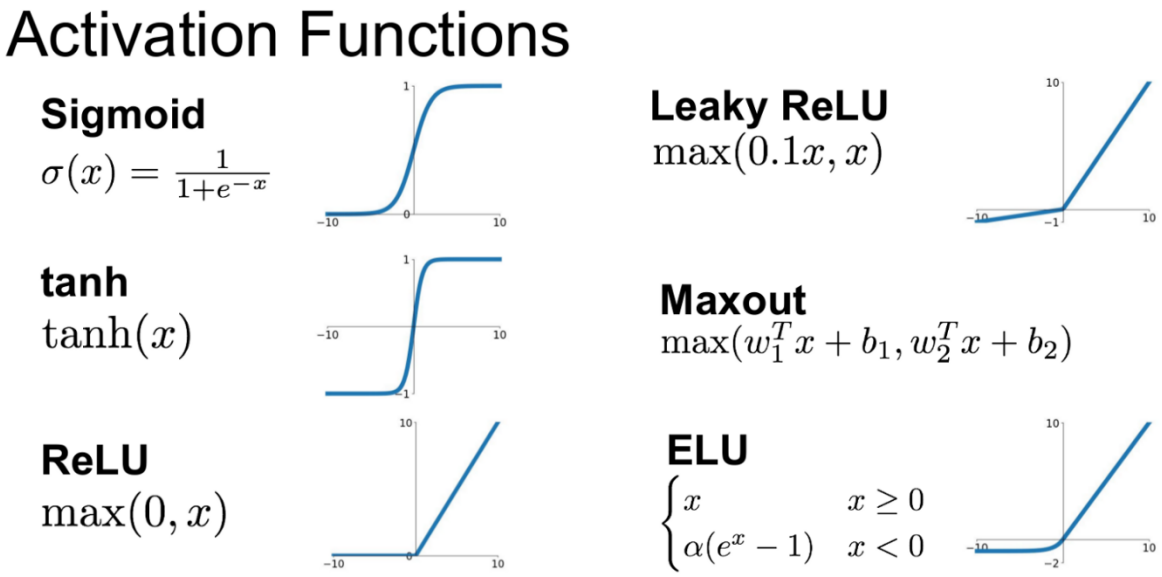
Реализация и обучение НС для задачи классификации

Была разработана нейронная сеть для классификации рака груди на основе признаков из датасета https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data?resource=download&select=data.csv

Для обучения нейронов в Keras используются различные функции активации, такие как:

- softmax,
- elu,
- relu,
- tanh,
- sigmoid,
- softsign.

Эти функции активации применяются через слои или аргументы активации в функциях Dense() и Activation().



Для создания нейронных сетей в Keras используют следующие слои:

1. Dense — основной слой, который вычисляет вывод с использованием активации, ядра и смещения.
2. Dropout — слой, применяемый для борьбы с переобучением, случайным образом устанавливает некоторые единицы в 0.
3. Flatten — слой, преобразующий данные в меньшую размерность для подачи на вход следующего слоя.
4. Input — слой, используемый для создания модели на основе ввода и вывода модели.
5. Reshape — слой, изменяющий форму вывода на заданную размерность.
6. Permute — слой, меняющий порядок измерений в данных.
7. Lambda — слой, позволяющий создавать дополнительные признаки, отсутствующие в Keras.
8. Conv1D и Conv2D — слои для создания свёрточных сетей.
9. MaxPooling1D и MaxPooling2D — слои для уменьшения размера ввода и извлечения важной информации.
10. AveragePooling1D и AveragePooling2D — слои для извлечения среднего значения из данных.
11. SimpleRNN и LSTM — рекуррентные слои для вычисления последовательных данных.

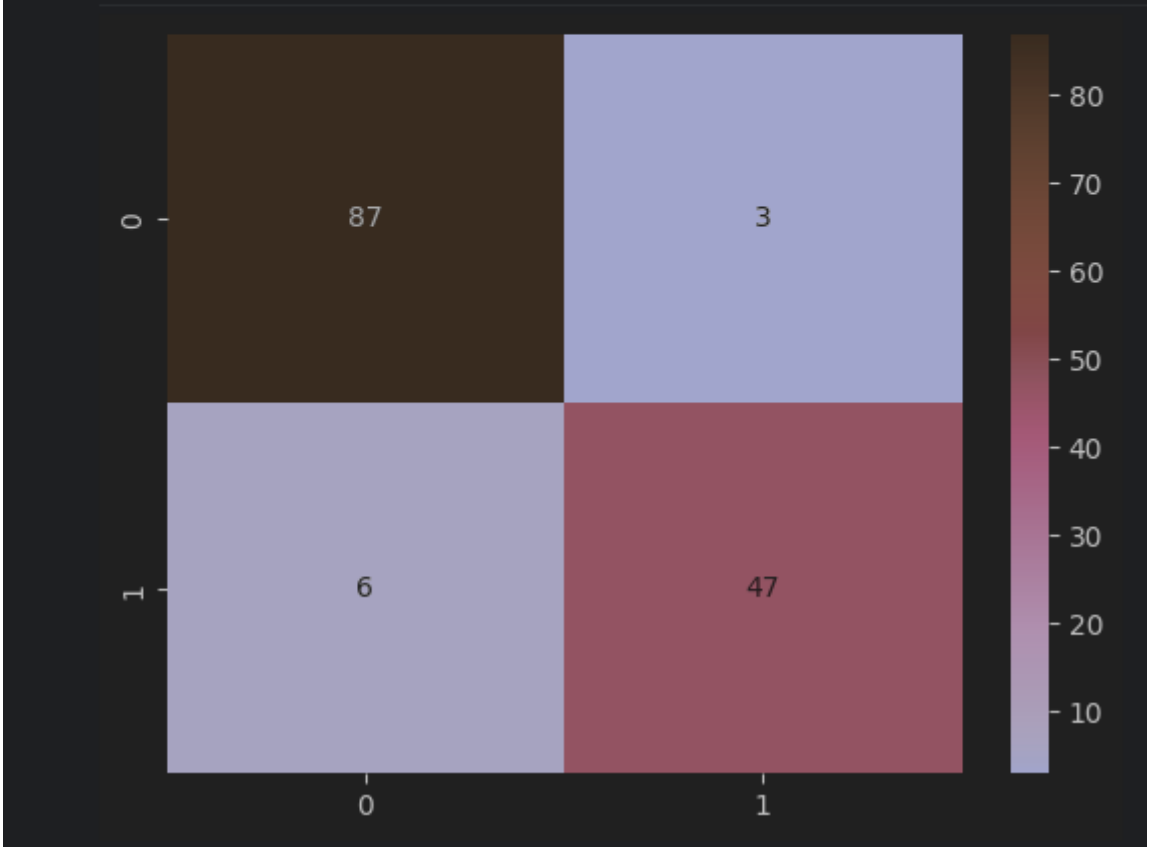
Гиперпараметры модели НС включают:

1. Количество скрытых слоев(layers) — определяет количество внутренних слоев в нейронной сети.

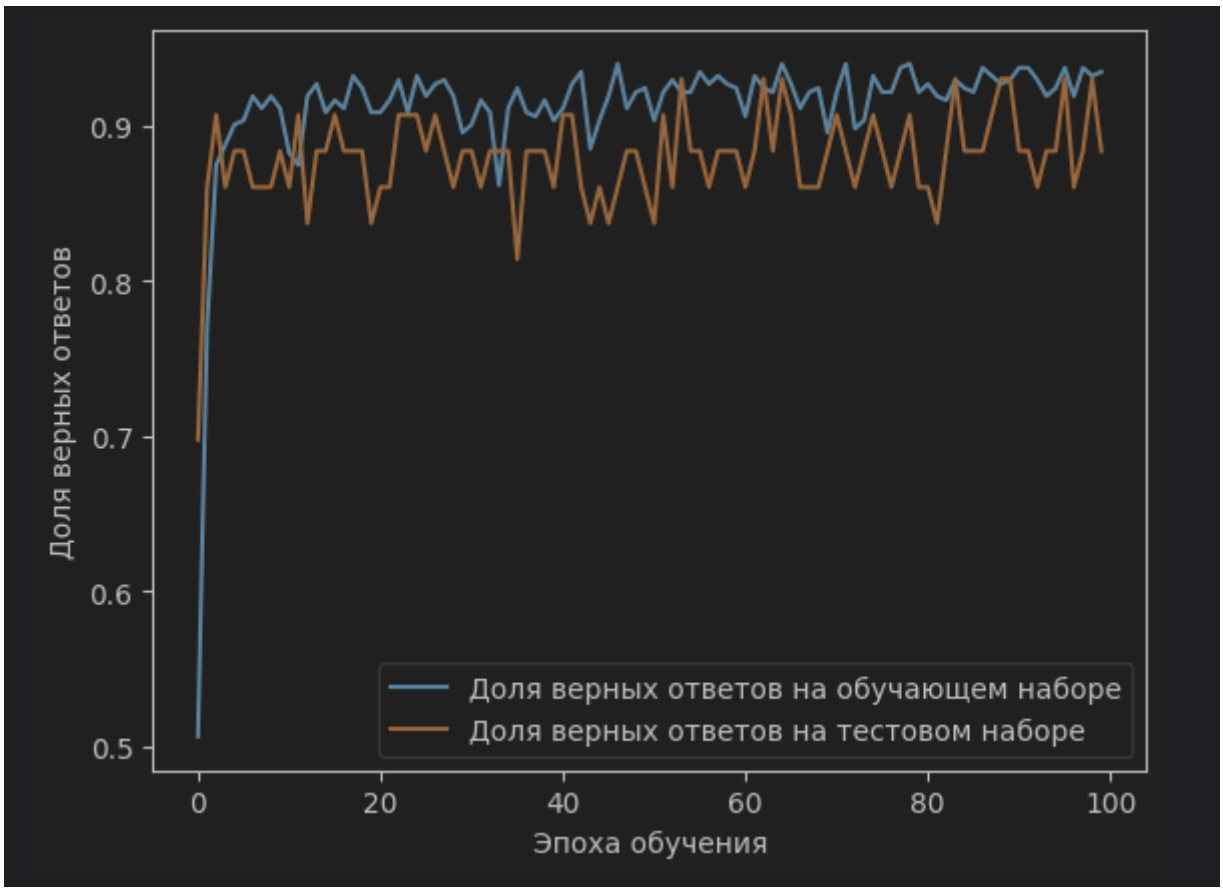
2. Количество нейронов в каждом скрытом слое(units) — указывает число узлов в каждом внутреннем слое.
3. Функции активации(activation) — выбирает функцию, которая преобразует выходные данные предыдущего слоя перед передачей их на следующий слой.
4. Скорость обучения(learning_rate) — определяет, насколько быстро модель будет обучаться на обучающих данных.
5. Коэффициент отсева(dropout) — контролирует вероятность того, что нейроны будут случайно отключены во время обучения.
6. Количество эпох(epoch) — указывает, сколько раз модель будет проходить через обучающие данные.
7. Размер батча (batch size) — это общее число тренировочных объектов, представленных в одном батче. Батчи используются для эффективного обучения больших объёмов данных, так как позволяют разделить данные на части меньшего размера и загружать их по очереди.
8. Оптимизатор(optimizer) — оптимизирует функцию потерь для нахождения наилучших весов для прогнозирования. В Keras доступны различные типы оптимизаторов, такие как SGD, RMSProp, Adam, Adadelta, Adagrad, Adamax, Nadam и FTRL.
9. Функция ошибки(loss) в контексте машинного обучения — это функция, которая оценивает качество работы модели и пытается минимизировать ошибку во время обучения.

Точность обученной модели:

	precision	recall	f1-score	support
0	0.94	0.97	0.95	90
1	0.94	0.89	0.91	53
accuracy			0.94	143
macro avg	0.94	0.93	0.93	143
weighted avg	0.94	0.94	0.94	143



Время выполнения: 5.43s

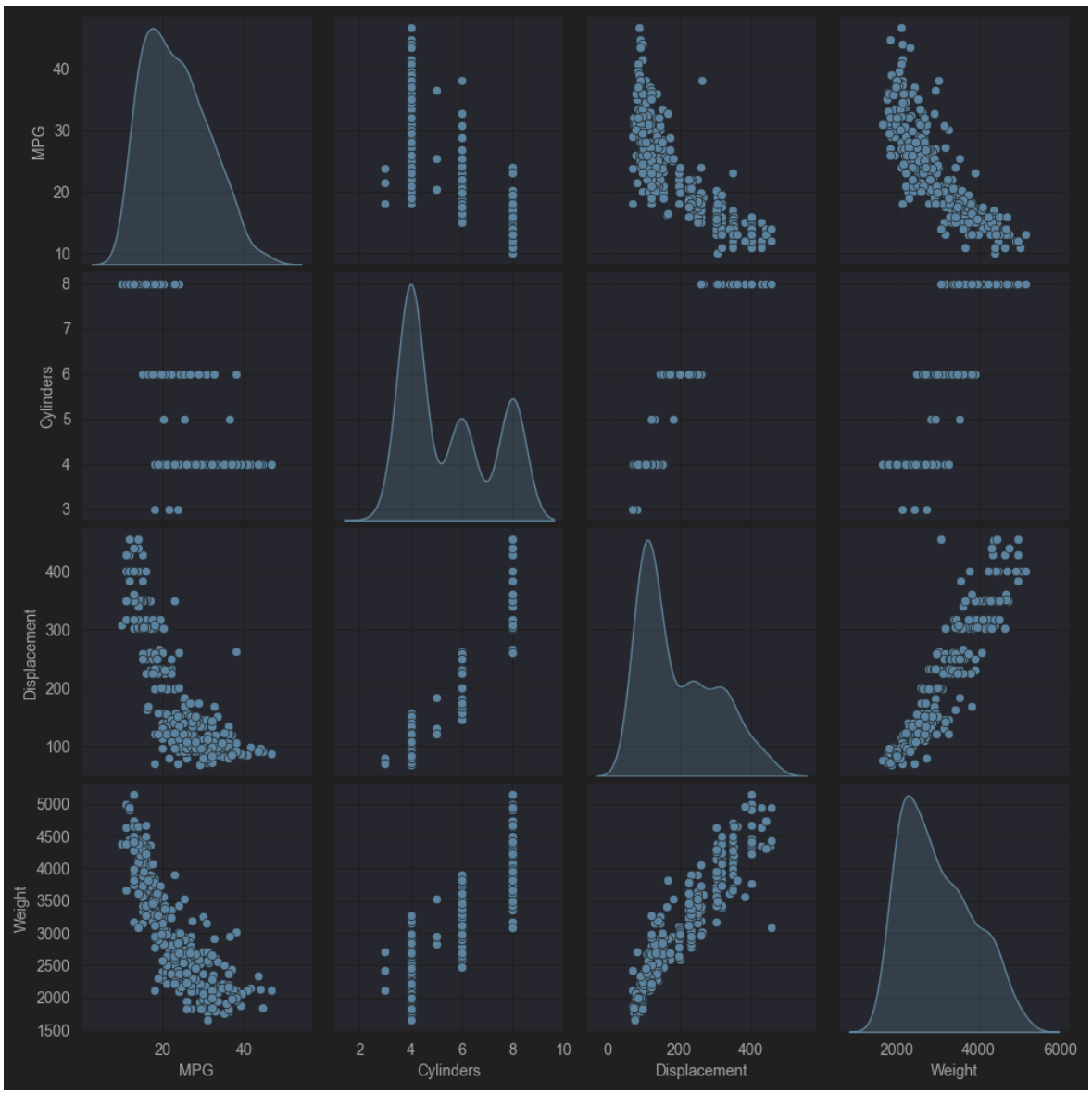


Реализация и обучение НС для задачи регрессии

Построены различные модели регрессии для прогнозирование эффективности использования топлива на основе признаков из датасета https://archive.ics.uci.edu/dataset/9/auto+mpg

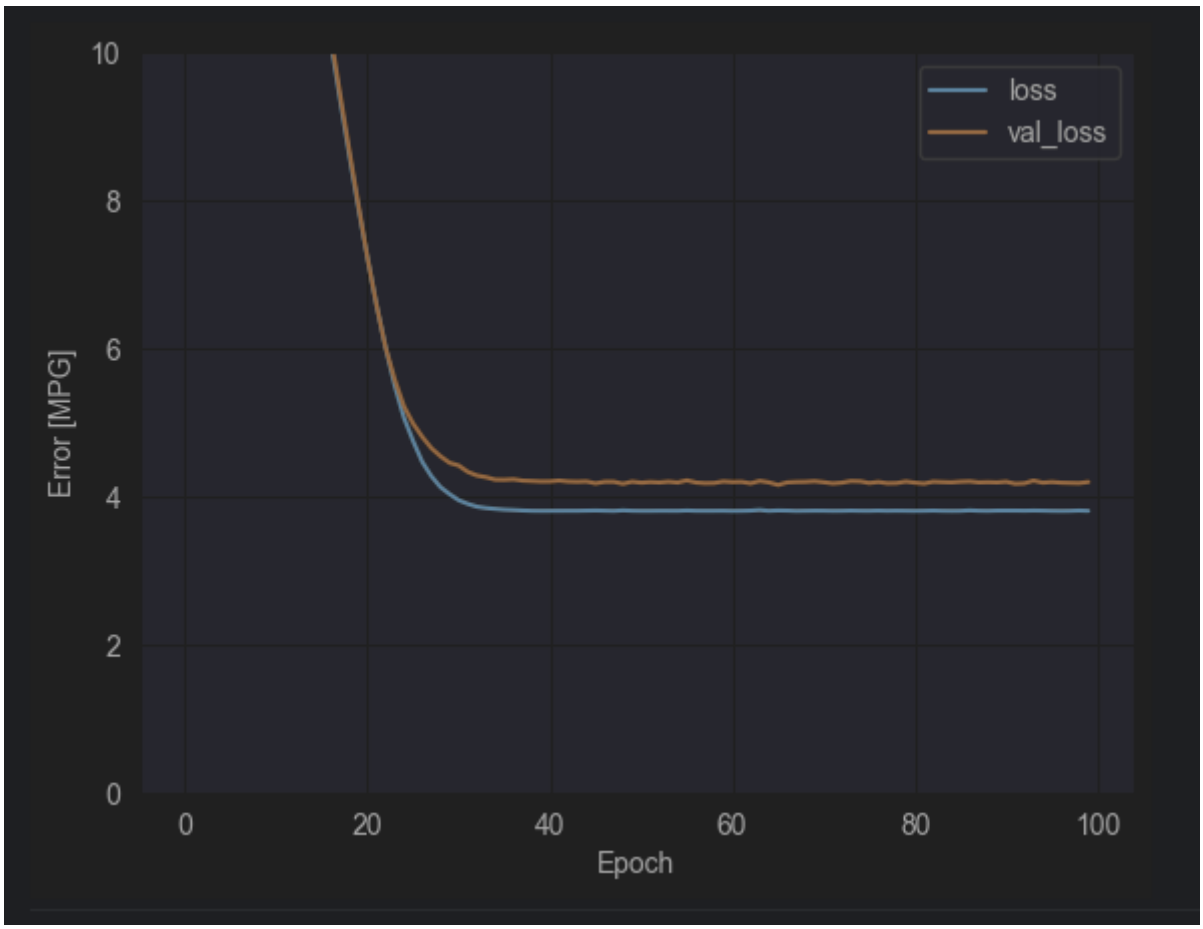
5 rows | 5 rows × 10 columns | Static Output

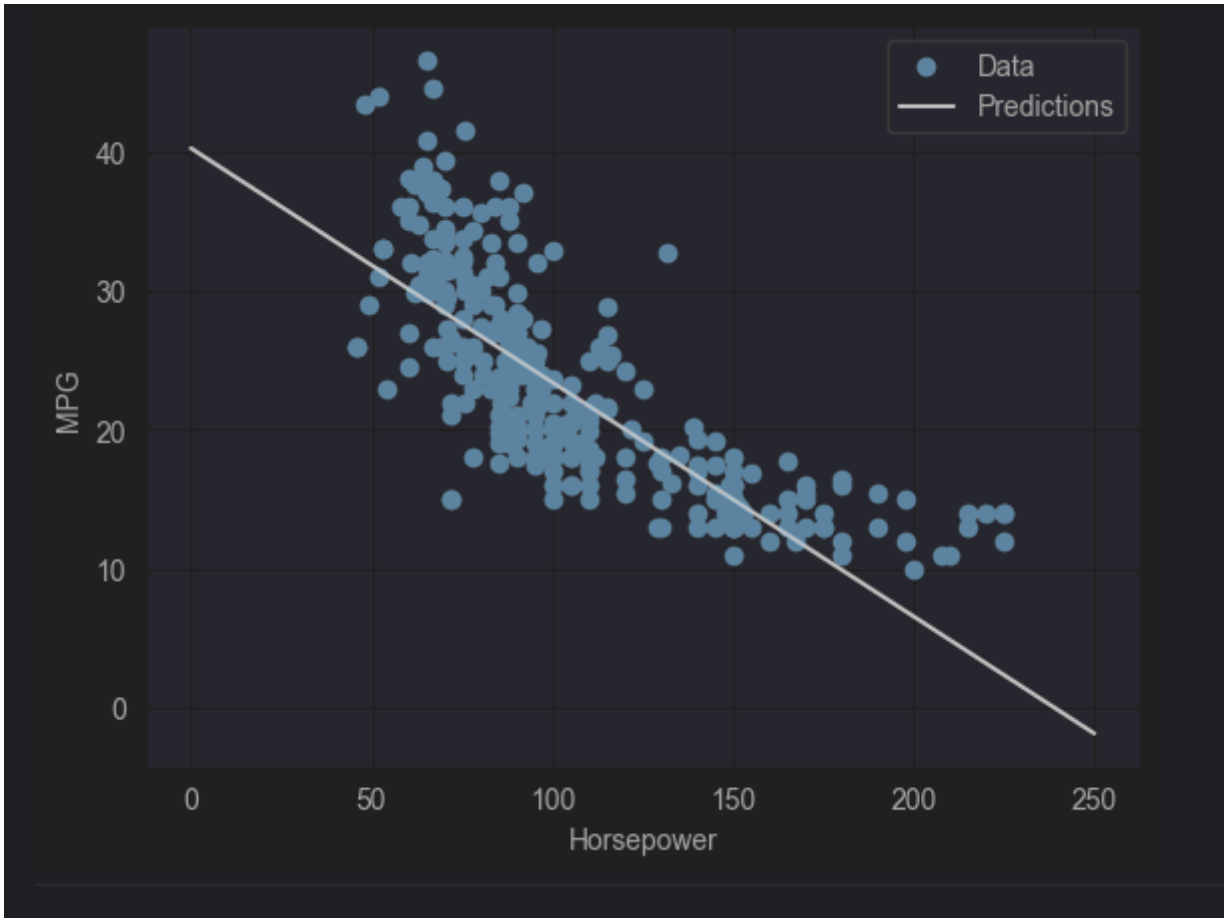
	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	Europe	Japan	USA
393	27.0	4	140.0	86.0	2790.0	15.6	82	False	False	True
394	44.0	4	97.0	52.0	2130.0	24.6	82	True	False	False
395	32.0	4	135.0	84.0	2295.0	11.6	82	False	False	True
396	28.0	4	120.0	79.0	2625.0	18.6	82	False	False	True
397	31.0	4	119.0	82.0	2720.0	19.4	82	False	False	True



Линейная регрессия с одной переменной. Предсказание `MPG` по `Horsepower`

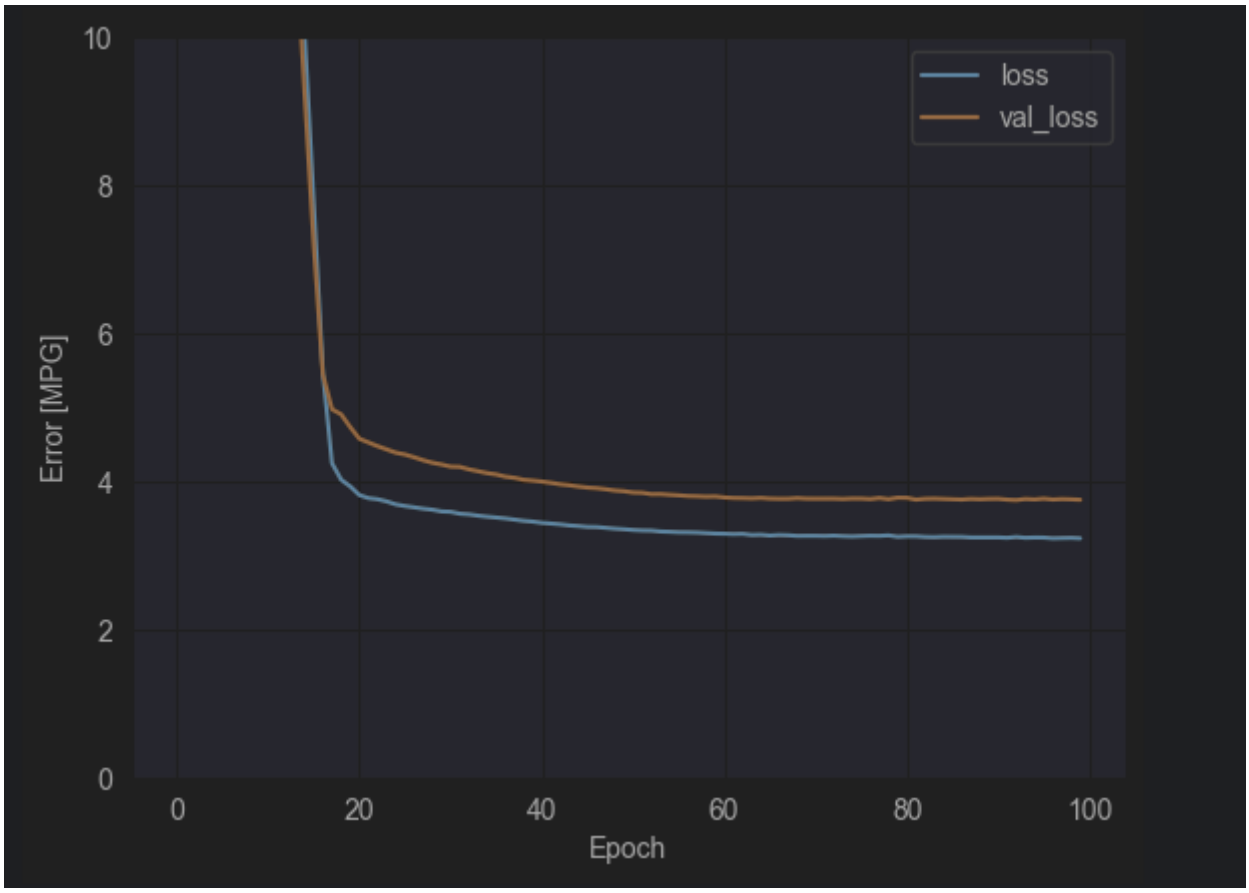
Layer (type)	Output Shape	Param #
normalization_2 (Normalization)	(None, 1)	3
dense_2 (Dense)	(None, 1)	2

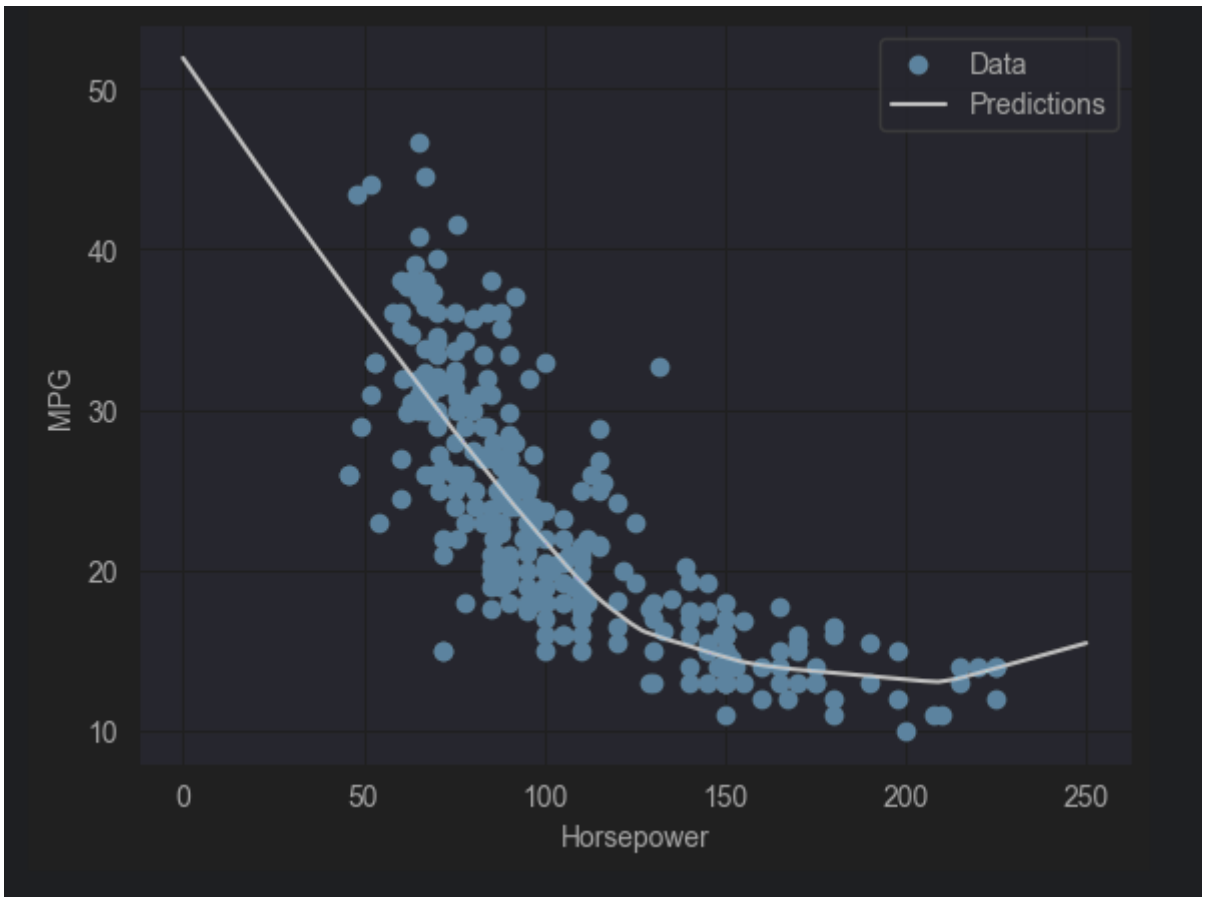

loss
val_loss
Error [MPG]
Epoch
0
2
4
6
8
10
0
20
40
60
80
100


Data
Predictions
MPG
Horsepower
0
10
20
30
40
0
50
100
150
200
250

Регрессия с глубокой нейронной сетью (DNN)

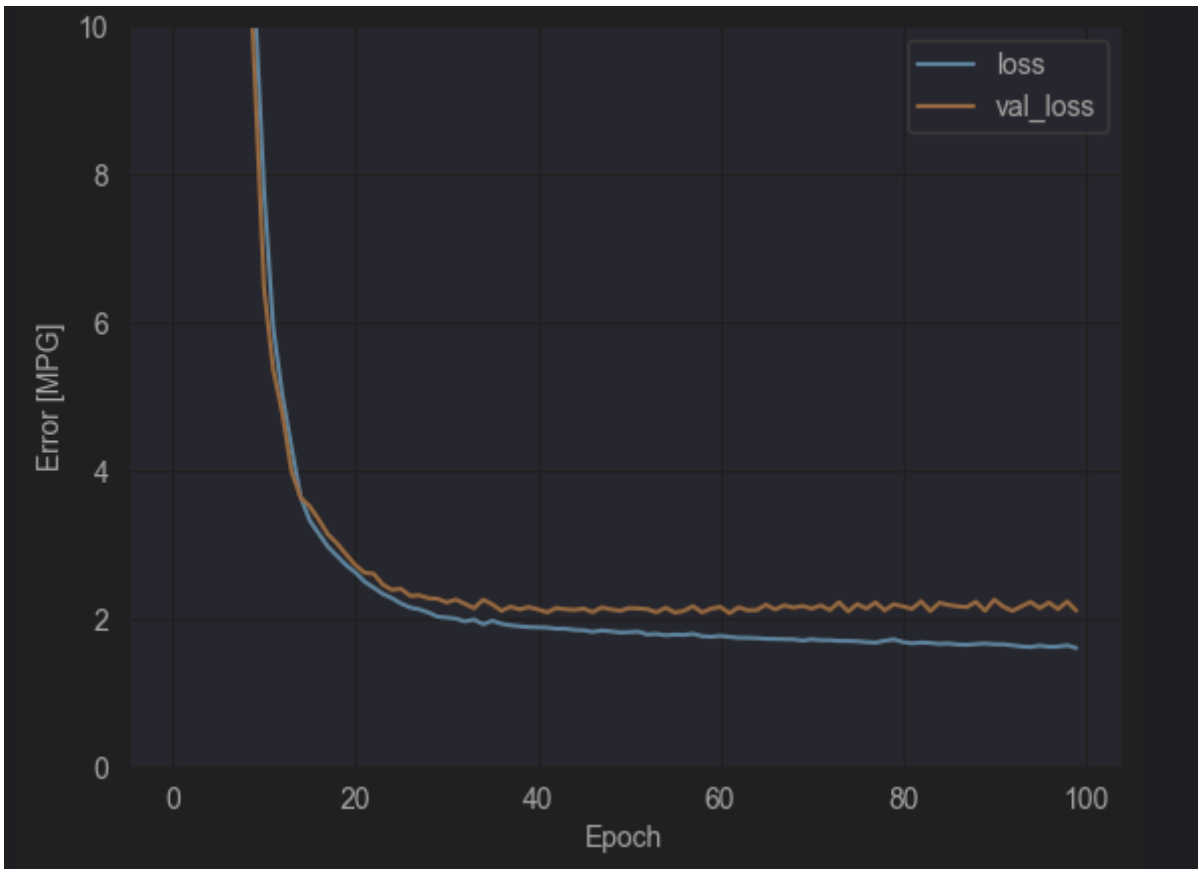
Layer (type)	Output Shape	Param #
normalization_2 (Normalization)	(None, 1)	3
dense_4 (Dense)	(None, 64)	128
dense_5 (Dense)	(None, 64)	4,160
dense_6 (Dense)	(None, 1)	65

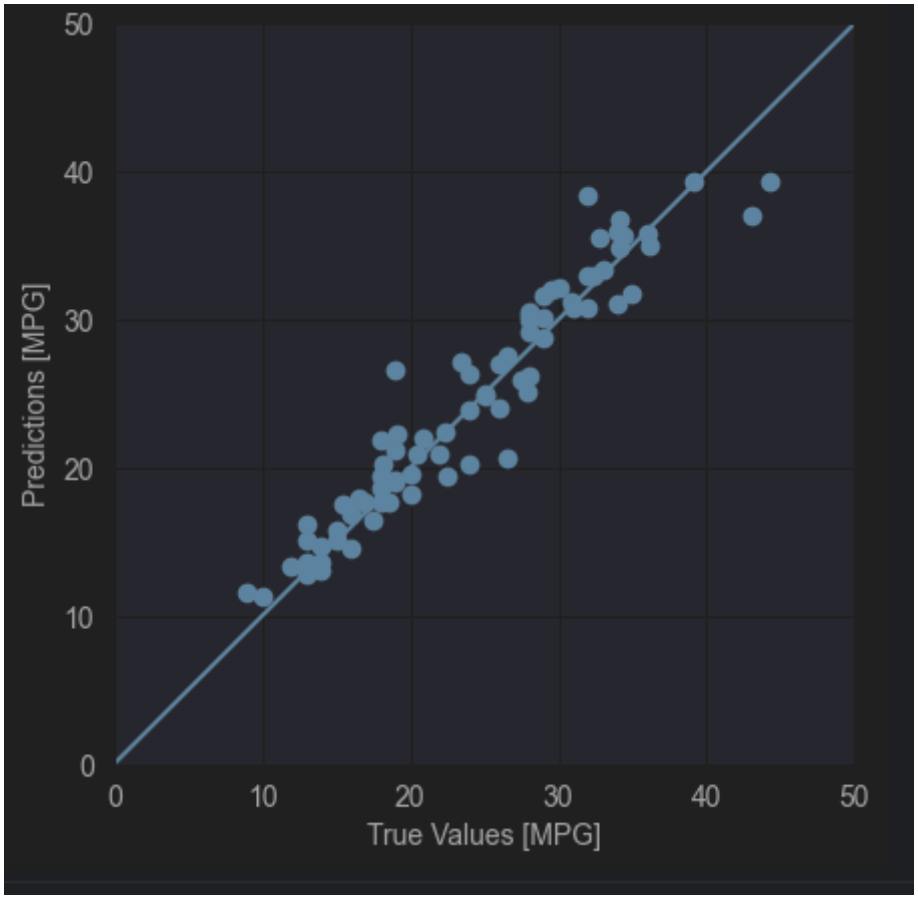




Регрессия с использованием DNN и нескольких входных данных

Layer (type)	Output Shape	Param #
normalization_1 (Normalization)	(None, 9)	19
dense_7 (Dense)	?	0 (unbuilt)
dense_8 (Dense)	?	0 (unbuilt)
dense_9 (Dense)	?	0 (unbuilt)

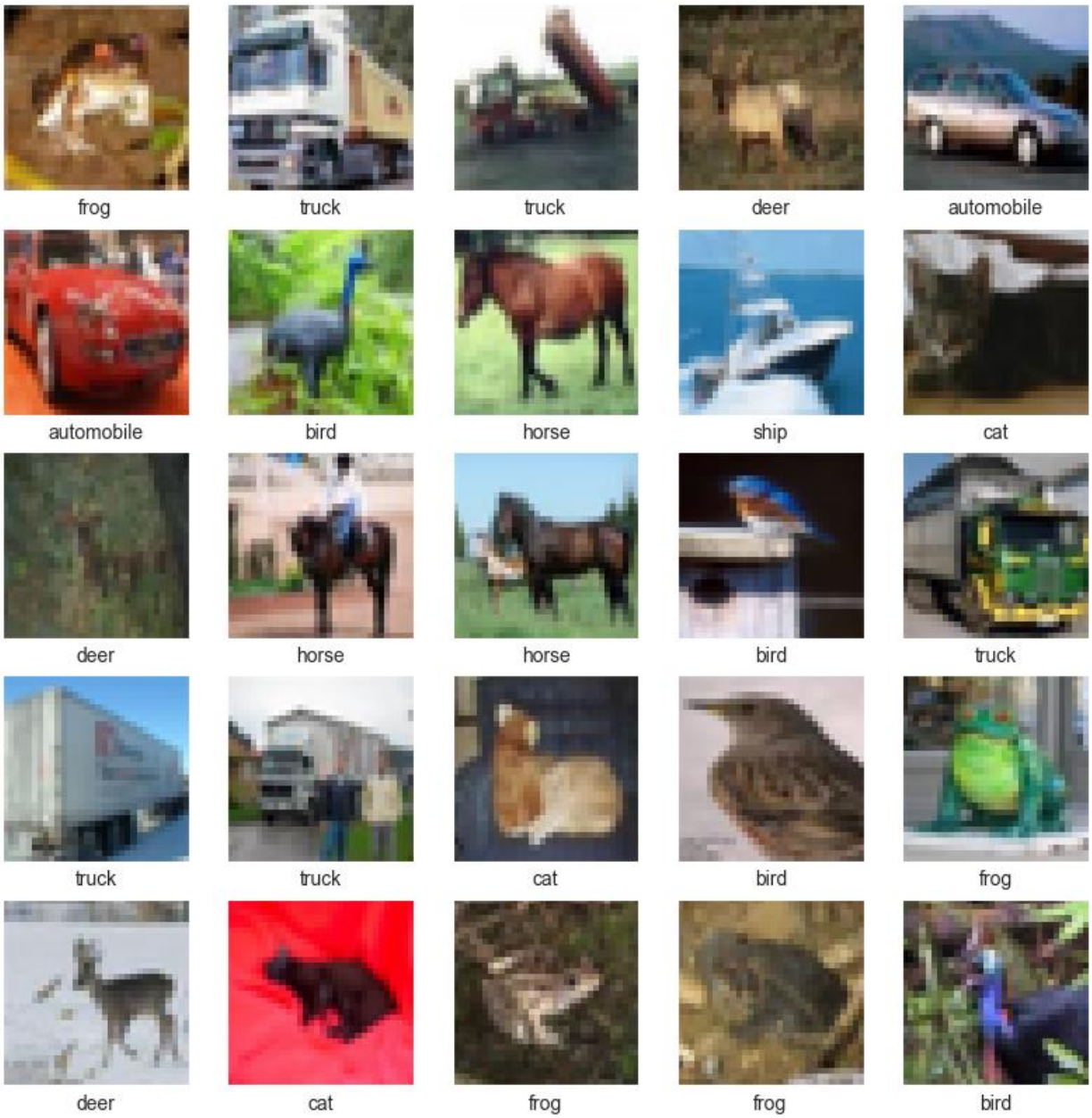

loss
val_loss
Error [MPG]
Epoch
0
2
4
6
8
10
0
20
40
60
80
100


Predictions [MPG]
True Values [MPG]
0
10
20
30
40
50

Реализация и обучение НС для распознавания изображений

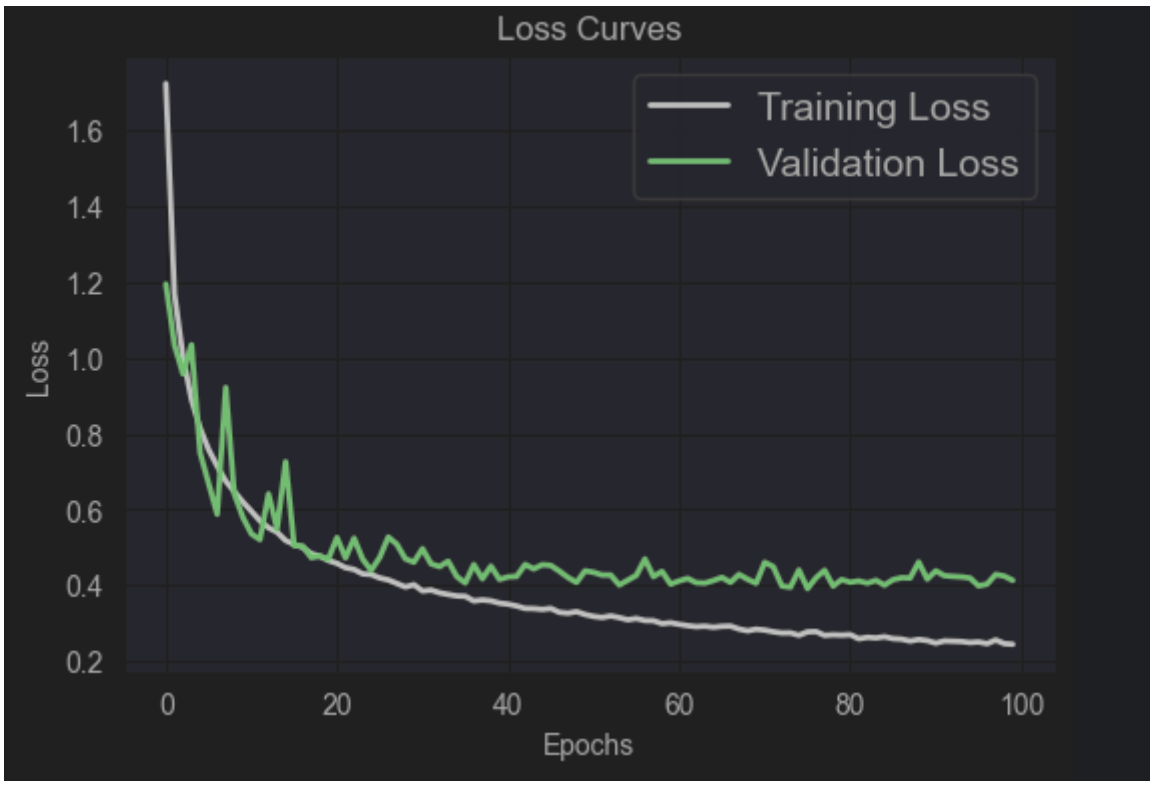
Реализована нейронная сеть для классификации изображений на основе датасетов CIFAR-10 и CIFAR-100

CIFAR-10

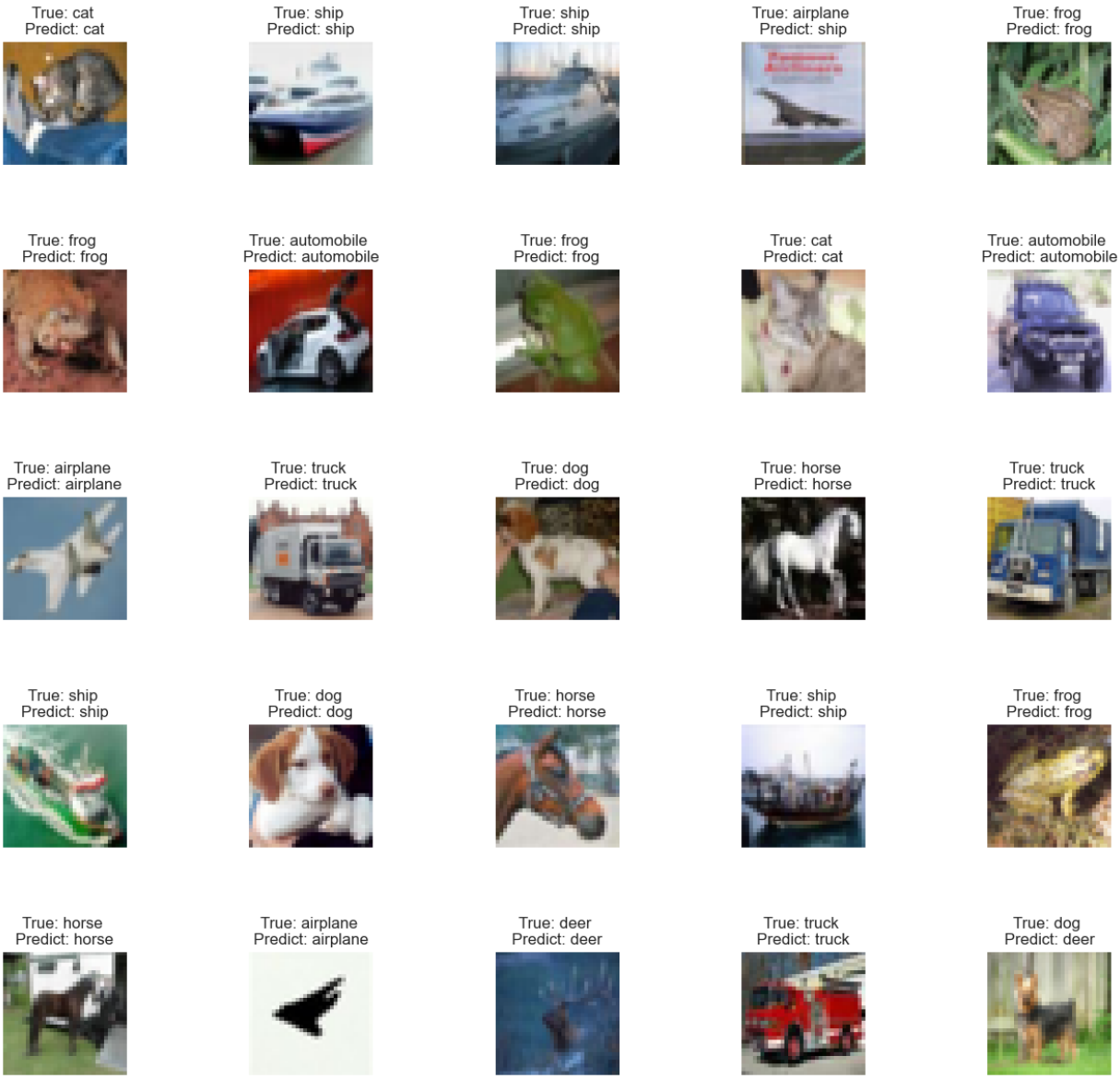


Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 32, 32, 32)	896
batch_normalization (BatchNormalization)	(None, 32, 32, 32)	128
conv2d_4 (Conv2D)	(None, 32, 32, 32)	9,248
batch_normalization_1 (BatchNormalization)	(None, 32, 32, 32)	128
max_pooling2d_2 (MaxPooling2D)	(None, 16, 16, 32)	0
dropout (Dropout)	(None, 16, 16, 32)	0
conv2d_5 (Conv2D)	(None, 16, 16, 64)	18,496
batch_normalization_2 (BatchNormalization)	(None, 16, 16, 64)	256
conv2d_6 (Conv2D)	(None, 16, 16, 64)	36,928
batch_normalization_3 (BatchNormalization)	(None, 16, 16, 64)	256
max_pooling2d_3 (MaxPooling2D)	(None, 8, 8, 64)	0
dropout_1 (Dropout)	(None, 8, 8, 64)	0
conv2d_7 (Conv2D)	(None, 8, 8, 128)	73,856
batch_normalization_4 (BatchNormalization)	(None, 8, 8, 128)	512
conv2d_8 (Conv2D)	(None, 8, 8, 128)	147,584
batch_normalization_5 (BatchNormalization)	(None, 8, 8, 128)	512
max_pooling2d_4 (MaxPooling2D)	(None, 4, 4, 128)	0
dropout_2 (Dropout)	(None, 4, 4, 128)	0
flatten_1 (Flatten)	(None, 2048)	0
dense_2 (Dense)	(None, 128)	262,272
batch_normalization_6 (BatchNormalization)	(None, 128)	512
dropout_3 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 10)	1,290

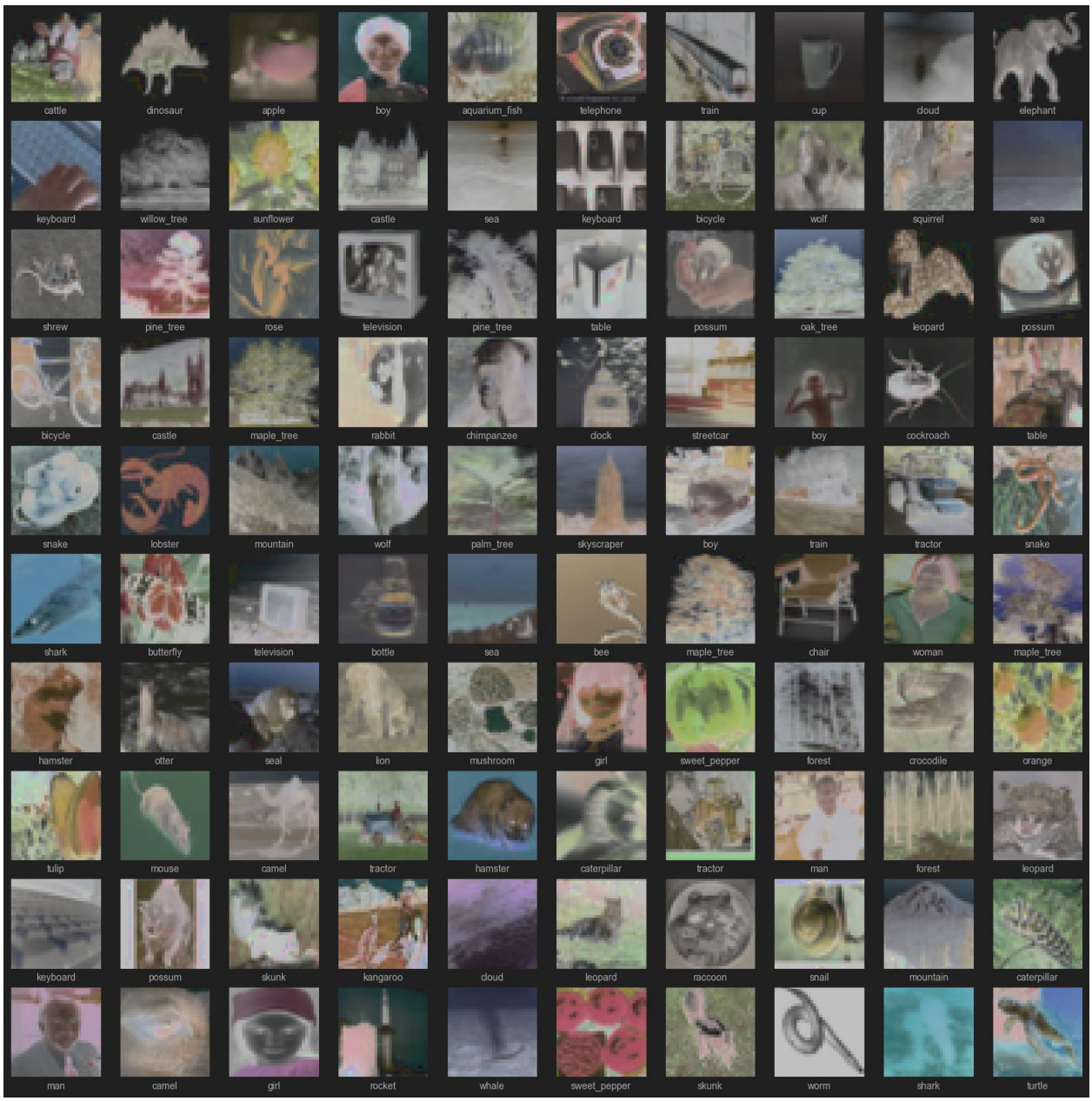
Время обучения: 16h 43min 8s



Предсказания:


True: cat
Predict: cat
True: ship
Predict: ship
True: ship
Predict: ship
True: airplane
Predict: ship
True: frog
Predict: frog
True: frog
Predict: frog
True: automobile
Predict: automobile
True: frog
Predict: frog
True: cat
Predict: cat
True: automobile
Predict: automobile
True: airplane
Predict: airplane
True: truck
Predict: truck
True: dog
Predict: dog
True: horse
Predict: horse
True: truck
Predict: truck
True: ship
Predict: ship
True: dog
Predict: dog
True: horse
Predict: horse
True: ship
Predict: ship
True: frog
Predict: frog
True: horse
Predict: horse
True: airplane
Predict: airplane
True: deer
Predict: deer
True: truck
Predict: truck
True: dog
Predict: deer

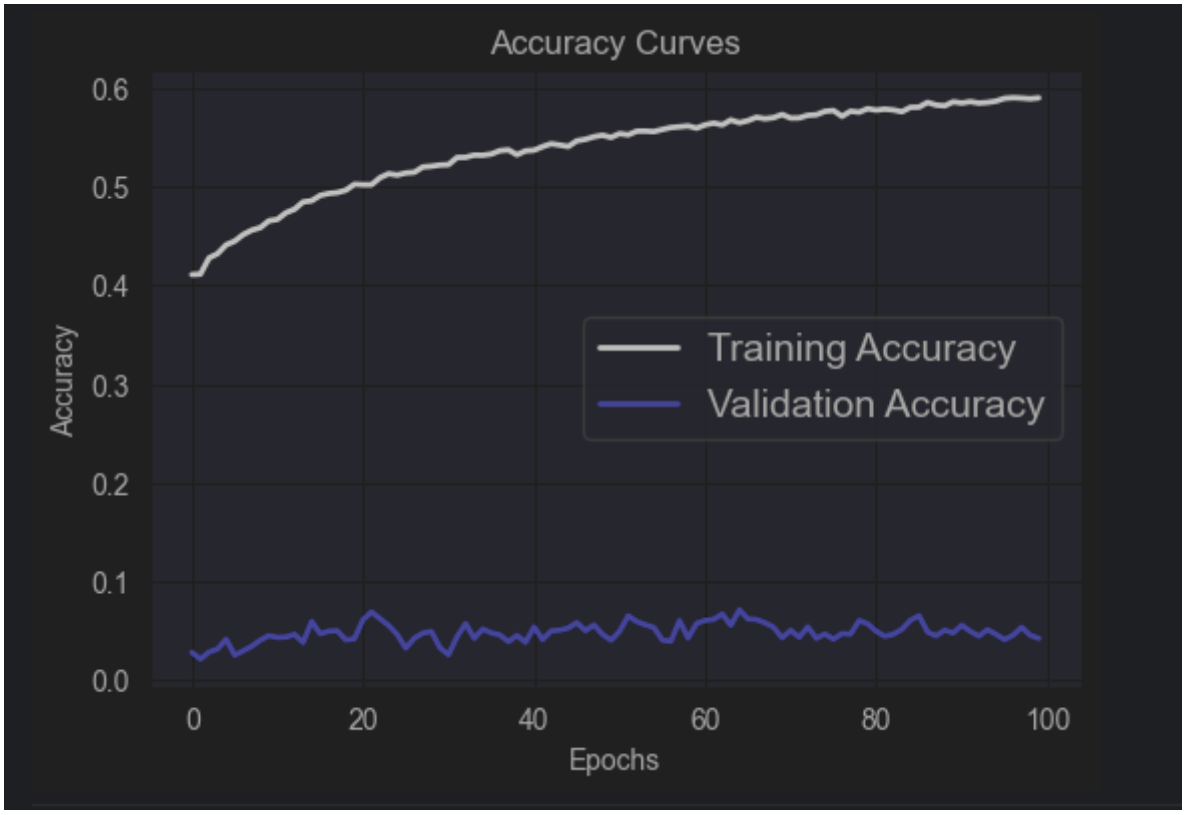
Cifar-100



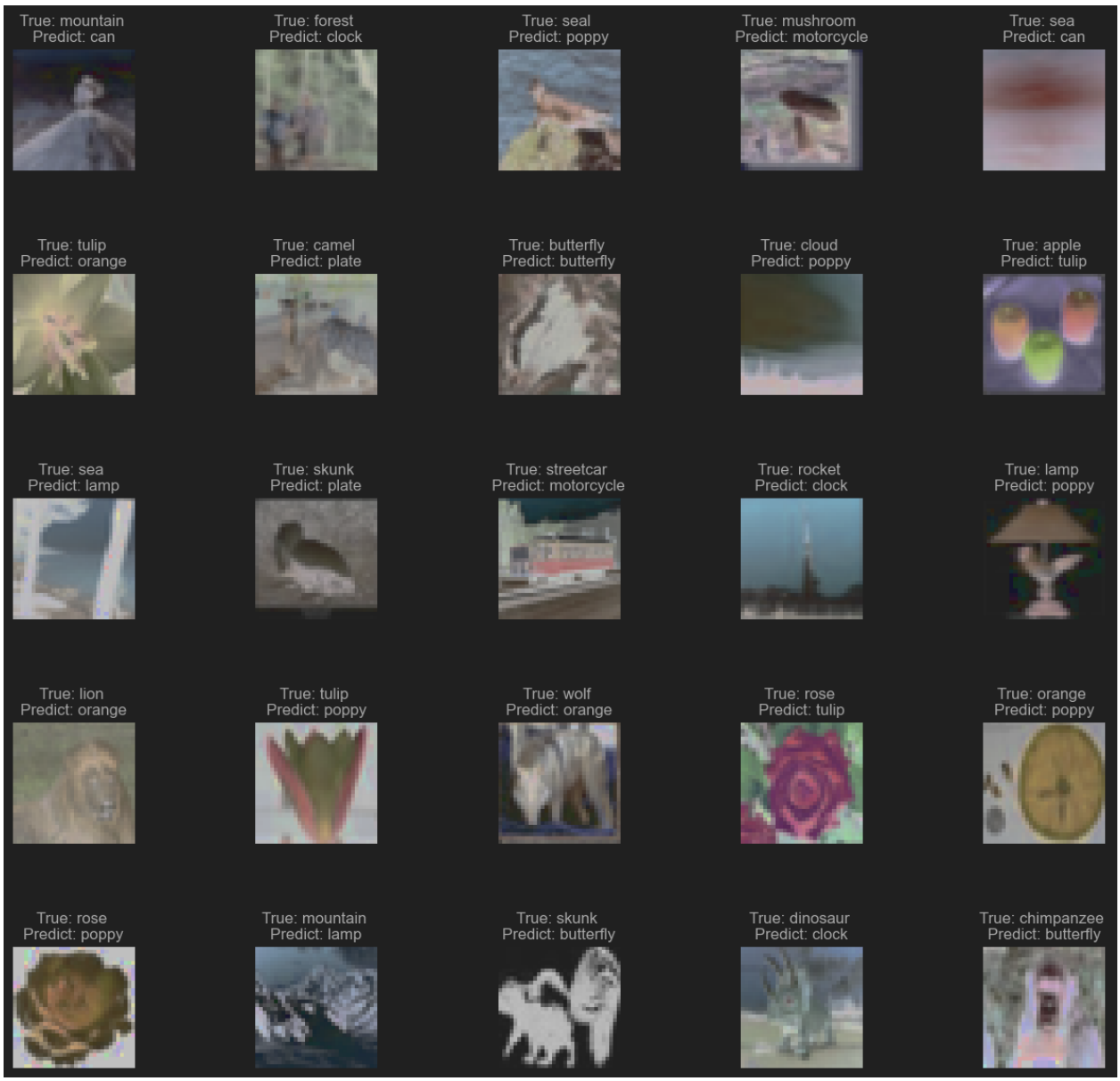

```
%%time
history = model.fit(train_images, train_labels, batch_size=64, epochs=100,
                    validation_data=(test_images, test_labels))
```

✓ [40] 1h 2m

```
Epoch 95/100
782/782 ━━━━━━━━━━━━━━━━━━━━ 38s 49ms/step - accuracy: 0.5908 - loss: 1.4431 - val_accuracy: 0.0461 - val_loss: 123.3379
Epoch 96/100
782/782 ━━━━━━━━━━━━━━━━━━━━ 38s 49ms/step - accuracy: 0.5920 - loss: 1.4413 - val_accuracy: 0.0407 - val_loss: 138.4801
Epoch 97/100
782/782 ━━━━━━━━━━━━━━━━━━━━ 38s 48ms/step - accuracy: 0.5922 - loss: 1.4234 - val_accuracy: 0.0452 - val_loss: 143.5919
Epoch 98/100
782/782 ━━━━━━━━━━━━━━━━━━━━ 38s 49ms/step - accuracy: 0.5928 - loss: 1.4348 - val_accuracy: 0.0533 - val_loss: 112.7119
Epoch 99/100
782/782 ━━━━━━━━━━━━━━━━━━━━ 39s 49ms/step - accuracy: 0.5910 - loss: 1.4457 - val_accuracy: 0.0451 - val_loss: 117.9640
Epoch 100/100
782/782 ━━━━━━━━━━━━━━━━━━━━ 38s 48ms/step - accuracy: 0.5909 - loss: 1.4363 - val_accuracy: 0.0420 - val_loss: 117.8785
CPU times: total: 16h 44min 16s
Wall time: 1h 2min 44s
```



Предсказания:


True: mountain
Predict: can
True: forest
Predict: clock
True: seal
Predict: poppy
True: mushroom
Predict: motorcycle
True: sea
Predict: can
True: tulip
Predict: orange
True: camel
Predict: plate
True: butterfly
Predict: butterfly
True: cloud
Predict: poppy
True: apple
Predict: tulip
True: sea
Predict: lamp
True: skunk
Predict: plate
True: streetcar
Predict: motorcycle
True: rocket
Predict: clock
True: lamp
Predict: poppy
True: lion
Predict: orange
True: tulip
Predict: poppy
True: wolf
Predict: orange
True: rose
Predict: tulip
True: orange
Predict: poppy
True: rose
Predict: poppy
True: mountain
Predict: lamp
True: skunk
Predict: butterfly
True: dinosaur
Predict: clock
True: chimpanzee
Predict: butterfly