

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ
по лабораторной работе №1
по дисциплине «Искусственные нейронные сети»
Тема: Многоклассовая классификация цветов

Студент гр. 8382

Преподаватель

Мирончик П.Д.

Жангиров Т.Р.

Санкт-Петербург

2021

ЗАДАНИЕ

Реализовать классификацию сортов растения ирис (Iris Setosa - 0, Iris Versicolour - 1, Iris Virginica - 2) по четырем признакам: размерам пестиков и тычинок его цветков.

- Ознакомиться с задачей классификации
- Загрузить данные
- Создать модель ИНС в Keras
- Настроить параметры обучения
- Обучить и оценить модель

Требования:

1. Изучить различные архитектуры ИНС (Разное кол-во слоев, разное кол-во нейронов на слоях)
2. Изучить обучение при различных параметрах обучения (параметры ф-ций fit)
3. Построить графики ошибок и точности в ходе обучения
4. Выбрать наилучшую модель

ХОД РАБОТЫ

1. Написание программы

Подготовка данных:

```
dataframe = pandas.read_csv("iris.csv", header=None)

dataset = dataframe.values

X = dataset[:, 0:4].astype(float)
Y = dataset[:, 4]

encoder = LabelEncoder()
encoder.fit(Y)
encoded_Y = encoder.transform(Y)
dummy_y = to_categorical(encoded_Y)
```

Из файла iris.csv получается таблица с 5 колонками: первые 4 - признаки ириса, последний - классы, которым соответствуют признаки. При

помощи `LabelEncoder` последний столбец преобразуется к категориальному виду.

```
def execute_model(layers=[], epochs=75, batch_size=10,
validation_split=0.1):
    model = Sequential()
    model.add(Dense(4, activation='relu'))

    for layer in layers:
        model.add(layer)

    model.add(Dense(3, activation='softmax'))

    model.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])

    return model.fit(
        X, dummy_y,
        epochs=epochs,
        batch_size=batch_size,
        validation_split=validation_split
    ).historylayersTests1
```

`executeModel` - вспомогательная функция, позволяющая настроить параметры сети.

`layers` - массив скрытых слоев

`epochs, batch_size, validation_split` - количество эпох, размер батча и количество данных для проверки соответственно.

```
def plot_single_history(history, color="blue"):
    keys = ["loss", "accuracy", "val_loss", "val_accuracy"]
    titles = ["Loss", "Accuracy", "Val loss", "Val accuracy"]
    xlabels = ["epoch", "epoch", "epoch", "epoch"]
    ylabels = ["loss", "accuracy", "loss", "accuracy"]
    ylims = [3, 1.1, 3, 1.1]

    for i in range(len(keys)):
        plot.subplot(2, 2, i + 1)
        plot.title(titles[i])
        plot.xlabel(xlabels[i])
        plot.ylabel(ylabels[i])
        plot.gca().set_ylim([0, ylims[i]])
        plot.grid()
        values = history[keys[i]]
        plot.plot(range(len(values)), values, color=color)
```

Функция `plot_single_history` строит графики для конкретной истории обучения. При этом несколько вызовов этой функции дадут разные

графики на одном экране (для их идентификации предлагается использовать цвет).

```
def run_layers_tests(tests):  
    for i in range(len(tests)):  
        history = execute_model(layers=tests[i], epochs=100)  
        plot_single_history(history, layersColors[i])
```


`run_layers_tests` запускает обучение моделей с разными параметрами скрытых слоев. `tests` - массив тестов, каждый из которых представляет собой массив скрытых слоев и передается в `execute_model`.

```
def run_params_tests(tests, layers_test):  
    for i in range(len(tests)):  
        test = tests[i]  
        history = execute_model(layers=layers_test, epochs=100,  
batch_size=test["batch"],  
validation_split=test["validation_split"])  
        plot_single_history(history, layersColors[i])
```

`run_params_tests` похожа на `run_layers_tests`, но запускает ряд тестов для оставшихся параметров сети: `batch_size` и `validation_split`.

2. Исследование моделей

2.1. Начальная сеть

Название сети	Параметры сети	Слои
 ИНС 1	epochs=100 batch_size=10 validation_split=0.1	4 relu 3 softmax

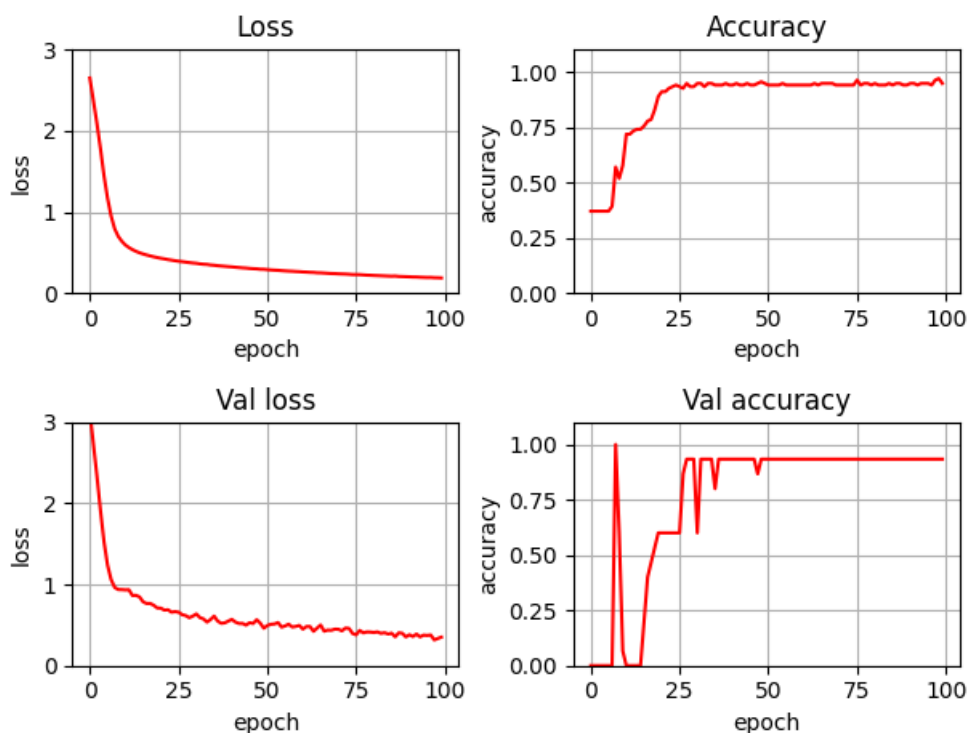


Рис.1, начальная ИНС

Заметно, что сеть ведет себя нестабильно. На данном примере модель быстро, примерно к 25 эпохе, обучается до точности, близкой к 1 (~0.93), как на данных для обучения, так и на проверочных данных. Однако далее ее обучение сильно замедляется, и к 100 эпохе точность остается на прежнем уровне (хотя показатели ошибок постепенно уменьшаются).

2.1. Исследование скрытых слоев

Для скрытых слоев будем использовать фиксированные параметры `epochs=100`, `batch_size=10` и `validation_split=0.1`.

Название сети	Слои
■ ИНС 2	4 relu 64 relu 3 softmax
■ ИНС 3	4 relu 192 relu 3 softmax

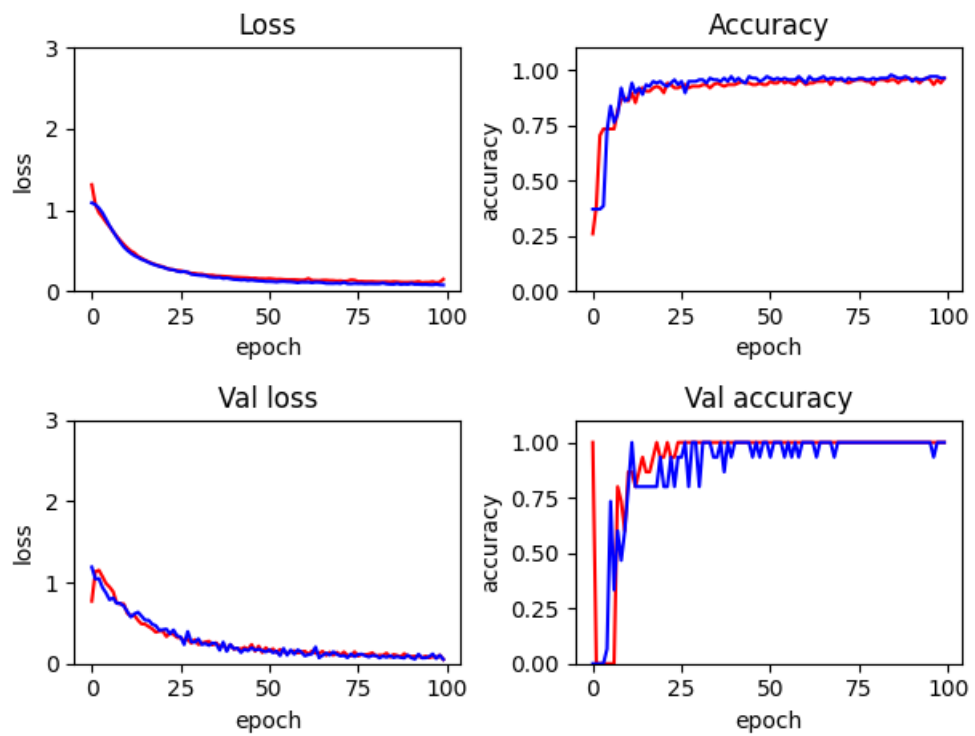


Рис.2, ИНС 2 и 3

Заметно, что увеличение количества нейронов на скрытом слое не дало ощутимой разницы в обучении сетей.

Название сети	Слои
ИНС 4	4 relu 64 sigmoid 3 softmax
ИНС 5	4 relu 192 sigmoid 3 softmax

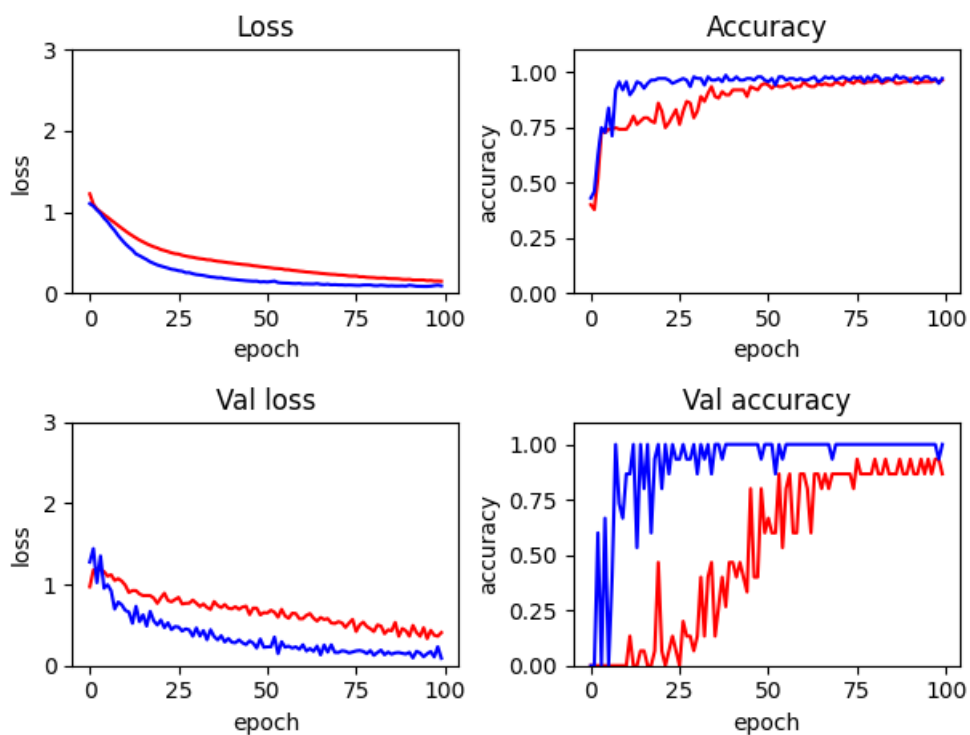


Рис.3, ИНС 4 и 5

Изменение функции активации на одном слое sigmoid также не дало ощутимых положительных результатов, однако заметно, что ИНС 5 со 192 нейронами на скрытом слое показывает значительно лучшие результаты в скорости обучения и точности по сравнению с ИНС 4.

Название сети	Слой
ИНС 6	4 relu 64 relu 64 relu 3 softmax
ИНС 7	4 relu 192 relu 192 sigmoid 3 softmax

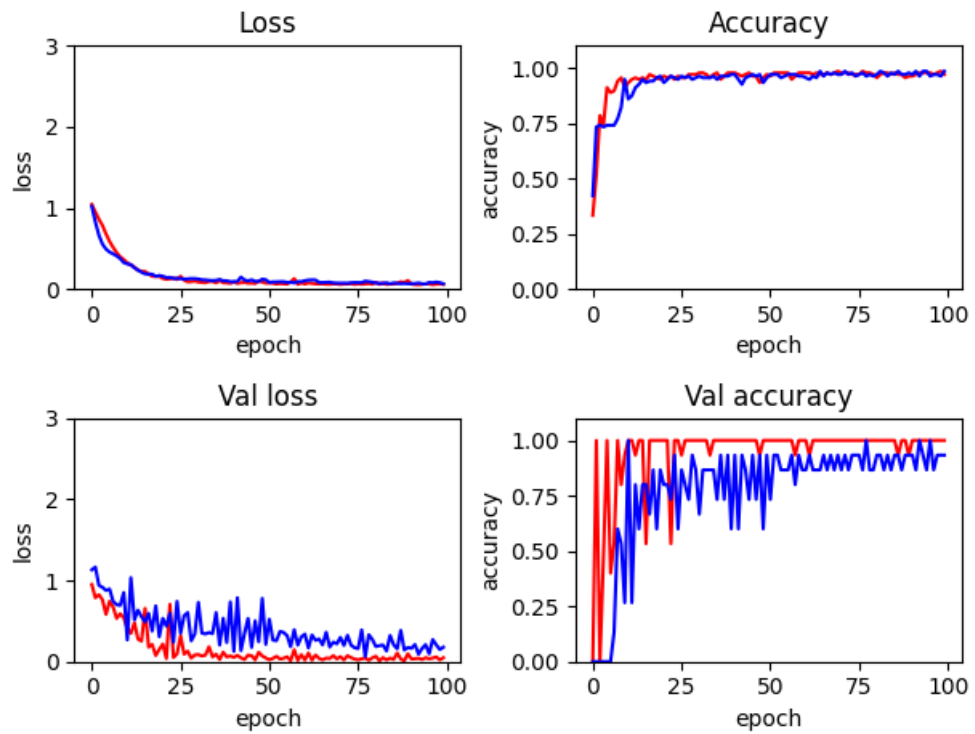


Рис. 4, ИНС 6 и 7

При добавлении второго скрытого слоя заметно возросла скорость обучения, количество потерь также значительно уменьшилось. При этом видно, что сеть с 64 нейронами на скрытых слоях показывает лучшие результаты, чем сеть со 192 нейронами.

Название сети	Слои
ИНС 8	4 relu 64 relu 64 relu 3 softmax
ИНС 9	4 relu 64 relu 64 relu 64 relu 3 softmax

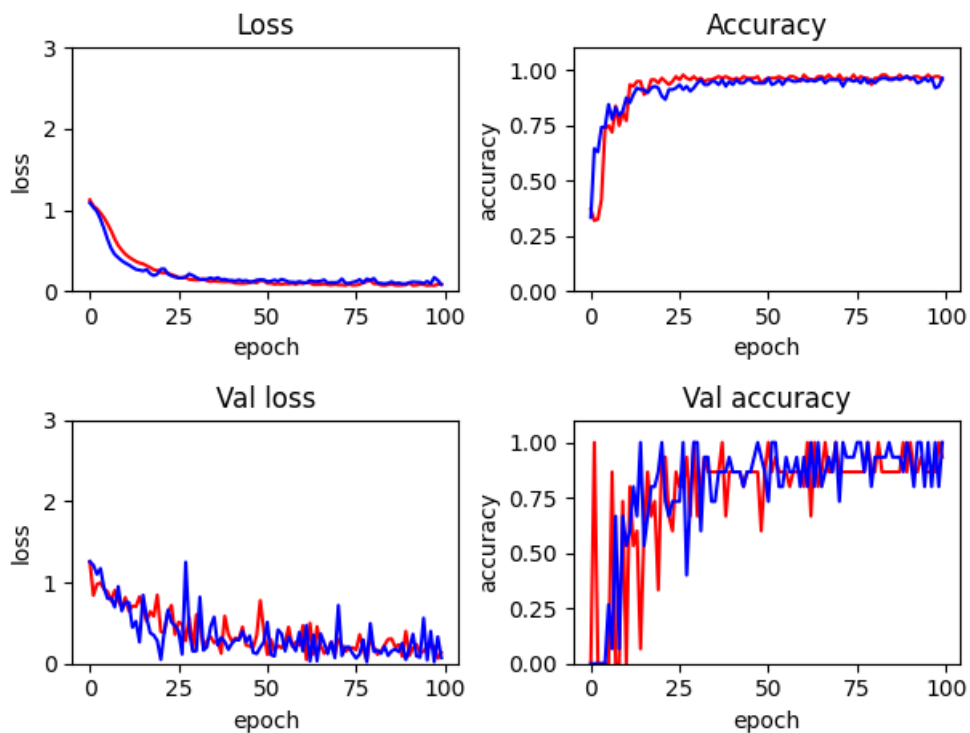


Рис. 5, ИНС 8 и 9

Добавление 3 слоя не сильно повлияло на результаты обучения.

2.3. Исследование параметров.

Далее будем использовать слои из ИНС 6, показавшей лучший результат.

```
4 relu
64 relu
64 relu
3 softmax
```

Название сети	Параметры
■ ИНС 10	batch_size: 10 validation_split: 0.1
■ ИНС 11	batch_size: 30 validation_split: 0.1
■ ИНС 12	batch_size: 10 validation_split: 0.3
■ ИНС 13	batch_size: 30 validation_split: 0.3

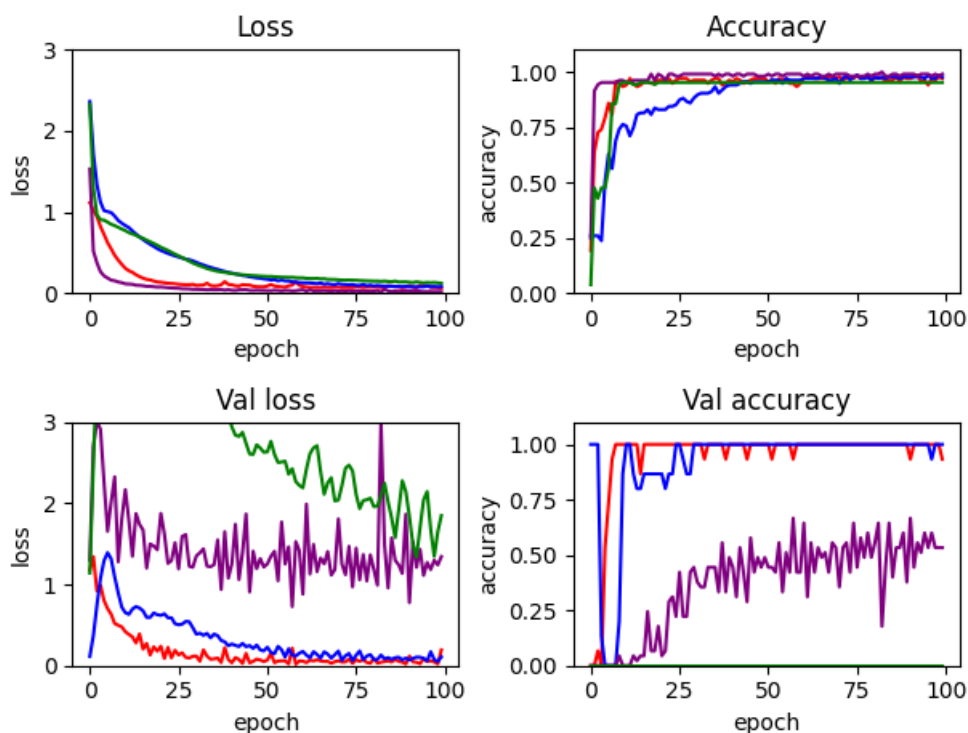


Рис. 6, ИНС 10, 11, 12 и 13

Видно, что лучшие результаты показали ИНС 10 и 11, имеющие параметр `validation_split=0.1`. ИНС 12 и 13 показывают хорошие результаты по потерям и точности на тестовых данных, но при этом ИНС 12 к 100 эпохе имеет количество потерь 1.4 и точность 0.5, а ИНС 13 - 1.9 и 0 соответственно. Возможно, сказывается маленькое количество исходных данных, и при большем количестве можно увидеть другую картину.

ВЫВОД

В ходе лабораторной работы были изучены основные параметры нейронных сетей, такие как скрытые слои, размер батча, количество эпох и размер проверочных данных. Был проведен анализ сетей с различными параметрами, проведено сравнение потерь и точности на тестовых и проверочных данных и по результатам исследований выбрана лучшая сеть для классификации цветков 3-х классов по 4-м параметрам.

ПРИЛОЖЕНИЕ А

Исходный код программы. main.py

```
import pandas

from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plot

def plot_single_history(history, color="blue"):
    keys = ["loss", "accuracy", "val_loss", "val_accuracy"]
    titles = ["Loss", "Accuracy", "Val loss", "Val accuracy"]
    xlabels = ["epoch", "epoch", "epoch", "epoch"]
    ylabels = ["loss", "accuracy", "loss", "accuracy"]
    ylims = [3, 1.1, 3, 1.1]

    for i in range(len(keys)):
        plot.subplot(2, 2, i + 1)
        plot.title(titles[i])
        plot.xlabel(xlabels[i])
        plot.ylabel(ylabels[i])
        plot.gca().set_ylim([0, ylims[i]])
        plot.grid()
        values = history[keys[i]]
        plot.plot(range(len(values)), values, color=color)

def execute_model(layers=[], epochs=75, batch_size=10,
validation_split=0.1):
    model = Sequential()
    model.add(Dense(4, activation='relu'))

    for layer in layers:
        model.add(layer)

    model.add(Dense(3, activation='softmax'))

    model.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])

    return model.fit(
        X, dummy_y,
        epochs=epochs,
        batch_size=batch_size,
        validation_split=validation_split
    ).history

def run_layers_tests(tests):
```

```

        for i in range(len(tests)):
            history = execute_model(layers=tests[i], epochs=100)
            plot_single_history(history, layersColors[i])

def run_params_tests(tests, layers_test):
    for i in range(len(tests)):
        test = tests[i]
        history = execute_model(layers=layers_test, epochs=100,
batch_size=test["batch"],
validation_split=test["validation_split"])
        plot_single_history(history, layersColors[i])

dataframe = pandas.read_csv("iris.csv", header=None)

dataset = dataframe.values

X = dataset[:, 0:4].astype(float)
Y = dataset[:, 4]

encoder = LabelEncoder()
encoder.fit(Y)
encoded_Y = encoder.transform(Y)
dummy_y = to_categorical(encoded_Y)

layersColors = [
    "red",
    "blue",
    "purple",
    "green",
]

# тесты количества нейронов relu
layersTests1Title = "relu neurons"
layersTests1 = [
    [
        Dense(64, activation='relu'),
    ],
    [
        Dense(192, activation='relu'),
    ],
]

# тесты количества нейронов softmax
layersTests2 = [
    [
        Dense(64, activation='sigmoid'),
    ],
    [
        Dense(192, activation='sigmoid'),
    ],
]

```

```

layersTests3 = [
    [
        Dense(64, activation='relu'),
        Dense(64, activation='relu'),
    ],
    [
        Dense(192, activation='relu'),
        Dense(192, activation='sigmoid'),
    ],
]

layersTests4 = [
    [
        Dense(64, activation='relu'),
        Dense(64, activation='relu'),
    ],
    [
        Dense(64, activation='relu'),
        Dense(64, activation='relu'),
        Dense(64, activation='relu'),
    ],
]

paramsTests = [
    {
        "batch": 10,
        "validation_split": 0.1,
    },
    {
        "batch": 30,
        "validation_split": 0.1,
    },
]

# исходная сеть
#run_layers_tests([[]])
#тест слоев
#run_layers_tests(layersTests2)
# тест параметров
#run_params_tests(paramsTests, layersTests3[0])

plot.show()

```