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

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Искусственные нейронные сети»

Тема: Бинарная классификация отраженных сигналов радара

Студент гр. 8382 _____ Нечепуренко Н.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Цель работы.

Реализовать классификацию между камнями (R) и металлическими цилиндрами (M) на основе данных об отражении сигналов радара от поверхностей.

60 входных значений показывают силу отражаемого сигнала под определенным углом. Входные данные нормализованы и находятся в промежутке от 0 до 1.

Задачи.

- Ознакомиться с задачей бинарной классификации
- Загрузить данные
- Создать модель ИНС в tf.Keras
- Настроить параметры обучения
- Обучить и оценить модель
- Изменить модель и провести сравнение. Объяснить результаты

Требования.

1. Изучить влияние кол-ва нейронов на слое на результат обучения модели.
2. Изучить влияние кол-ва слоев на результат обучения модели.
3. Построить графики ошибки и точности в ходе обучения.
4. Провести сравнение полученных сетей, объяснить результат.

Выполнение работы.

Построим модель, состоящую из двух слоёв: входного с 60 нейронами, так как во входных данных имеется 60 признаков, с функцией активации relu и выходного с 1 сигмоидальным нейроном.

Все модели в данной работе будем компилировать с оптимизатором adam, функцией потерь перекрестная бинарная энтропия и метрикой accuracy, так как классы сбалансированы.

Проводить обучения моделей будем тоже с постоянными параметрами, в течение 100 эпох, при размере батча в 10 наблюдений, и разбиении 90 на 10% исходных данных на тренировочные и валидационные.

Результаты первой архитектуры модели приведены на рисунке 1.

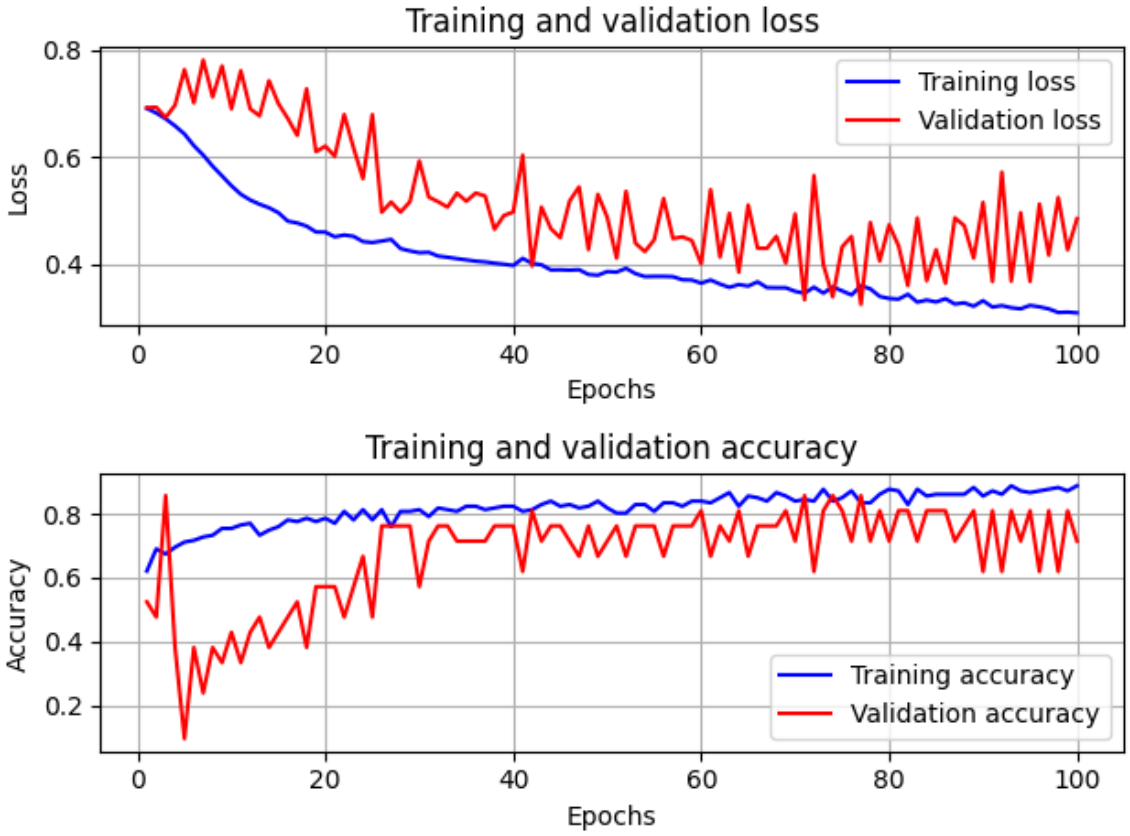

Рисунок 1 – Результат обучения первой архитектуры с двумя слоями

В результате трех запусков значение точности на валидационном множестве не превысило порога в 0.86.

Исходя из предположения об избыточности данных, сократим число рассматриваемых признаков вполовину, до 30, и изменим соответственно входной слой. Обучим модель (см. рис. 2).

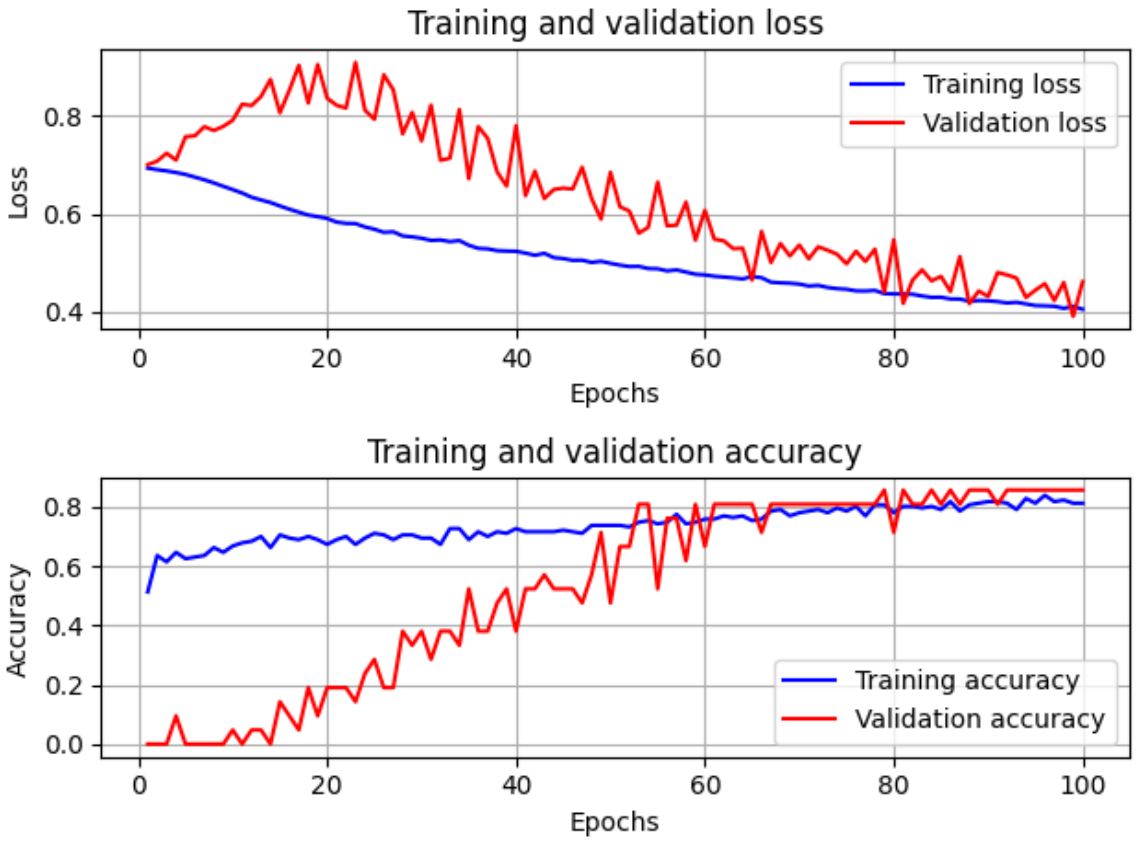

Рисунок 2 – Результат обучения второй архитектуры на 30 признаках

Данная модель обучается чуть медленней предыдущей, но имеет аналогичную точность. На основании полученных результатов можно сделать вывод о достоверности выдвинутого предположения об избыточности данных.

Добавим в модель скрытый полносвязный слой с функцией активации relu и 15 нейронами. В результате лучшего из пяти запусков были получены следующие результаты (см. рис. 3).

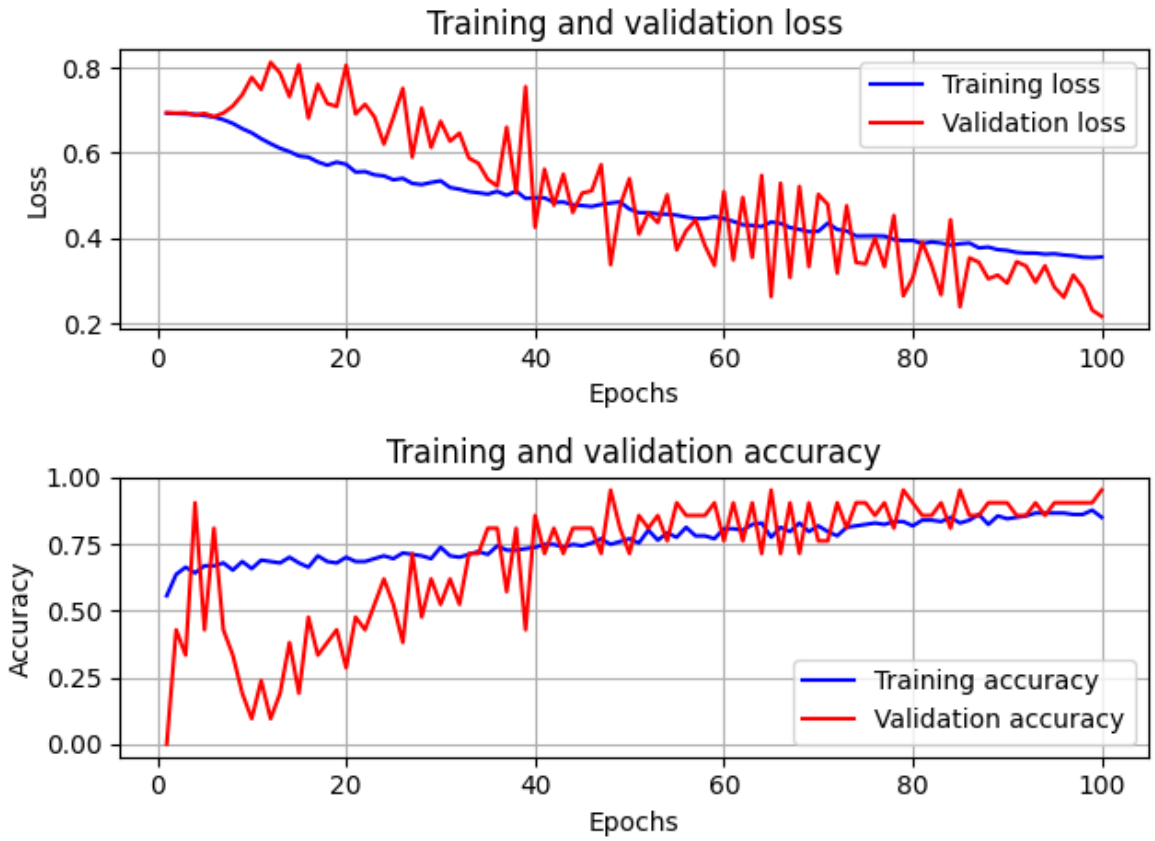

Рисунок 3 – Результат обучения модели с одним скрытым слоем

Заметим, что добавление скрытого слоя позволяет модели устанавливать более сложные закономерности во входных данных, что увеличивает точность. Получаем значение метрики accuracy в районе 0.92.

В исследовательских целях протестируем еще две архитектуры. Попробуем увеличить количество нейронов в скрытом слое до 30 в первом случае и добавим к этой модели второй слой в 15 нейронов во втором. Результаты приведены на рисунках 4 и 5 соответственно.

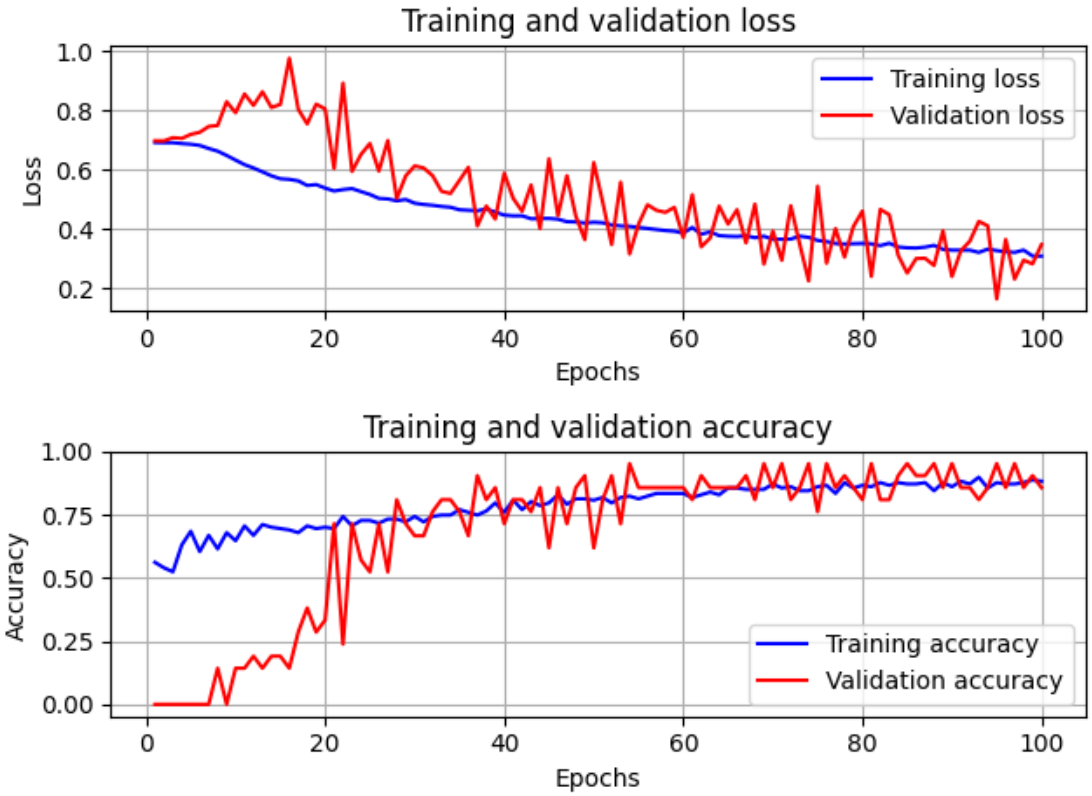

Рисунок 4 – Результат обучения архитектуры с 30 нейронами в скрытом слое

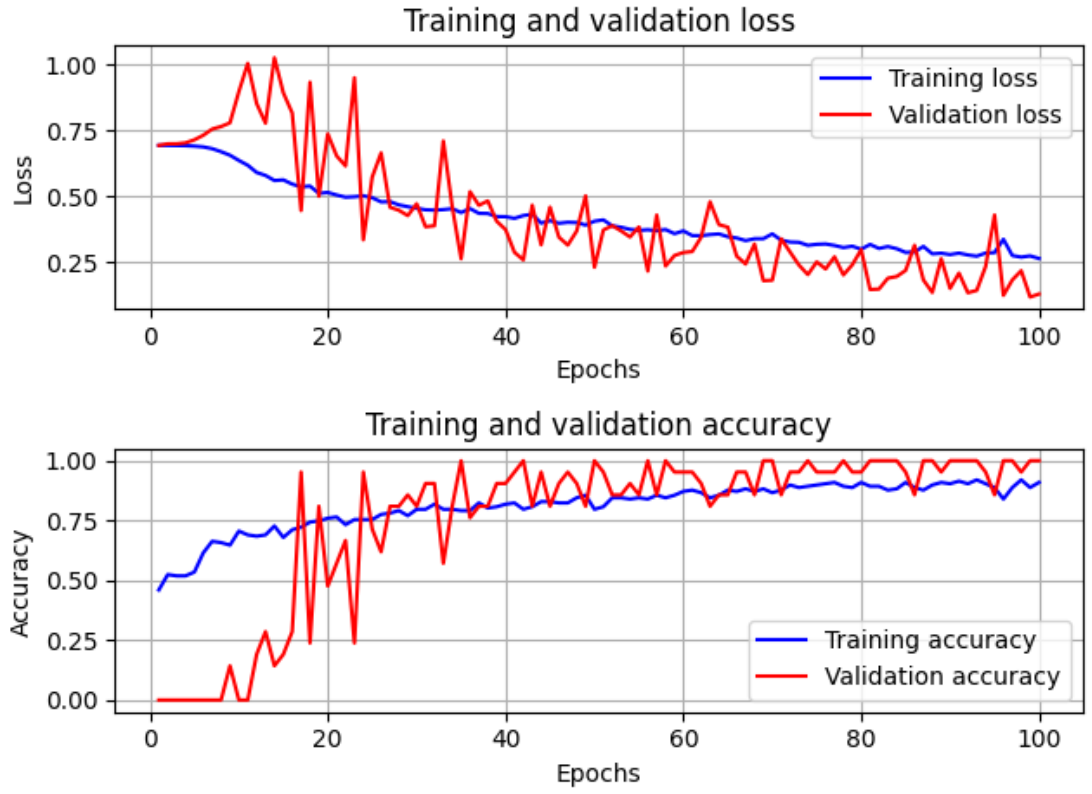

Рисунок 5 – Результат обучения архитектуры с 30 и 15 нейронами в скрытых слоях

Результаты двух последних архитектур немного лучше, но графики все равно довольно нестабильные.

Рассмотрим еще одну архитектуру. Вернемся к рассмотрению 60 признаков и добавим скрытый слой с 30 нейронами. Результаты модели приведены на рисунке 6.

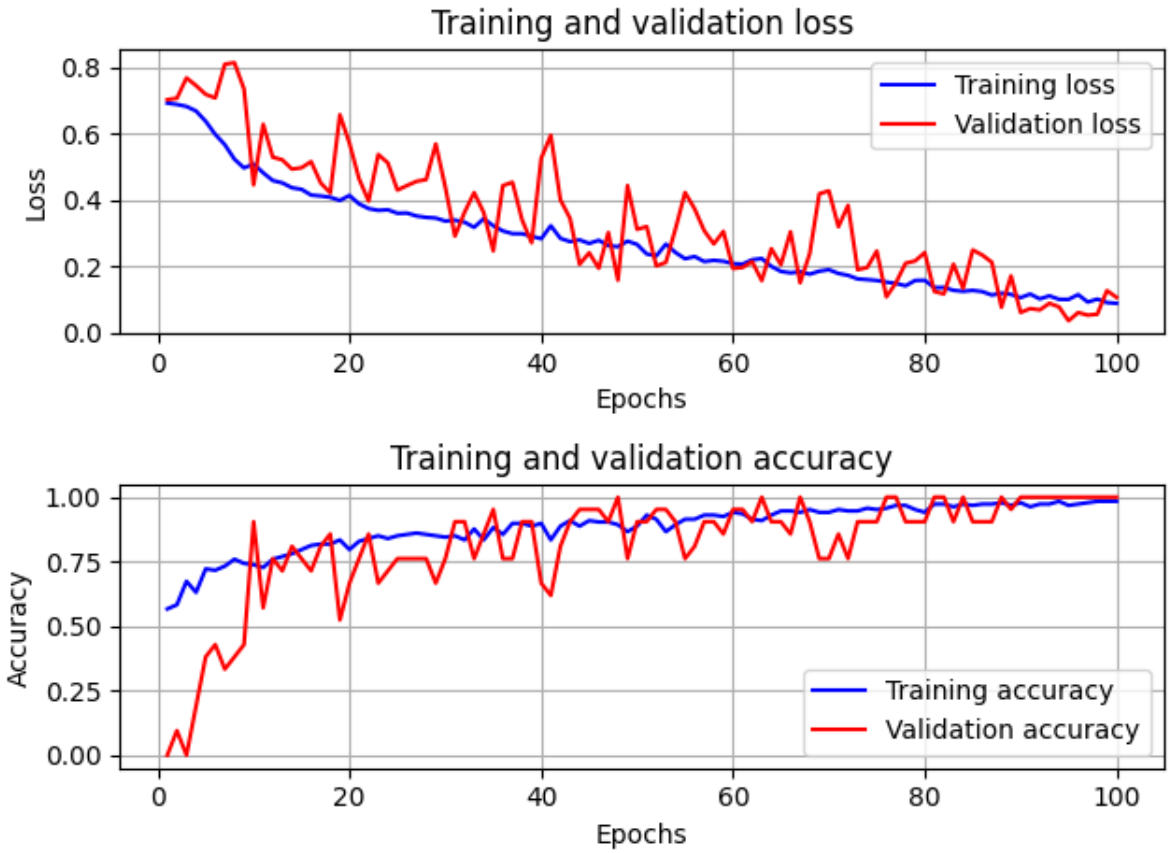

Рисунок 6 – Результат архитектуры с 60 признаками и скрытым слоем на 30 нейронов

К концу обучения графики точности на тренировочных и валидационных данных сходятся к 1, что может говорить о возможном переобучении модели. Исходный код программы приведен в Приложении А.

Выводы.

В ходе выполнения лабораторной работы были исследованы различные архитектуры ИНС для решения задачи бинарной классификации данных об отражении сигналов радара от поверхностей. Было изучено влияние количества

рассматриваемых признаков на качество модели, а также влияние количества нейронов на слое и, собственно, количества слоев. Архитектуры со скрытыми слоями показали более высокие результаты, чем архитектуры без них. Для проверки предположения об избыточности входных данных необходимо изучить описание признаков во входных данных, и, возможно отобрать их вручную, либо протестировать архитектуры с помощью более продвинутых методик, дабы уменьшить риск переобучения модели и отбросить лишние признаки, улучшив качество модели.

ПРИЛОЖЕНИЕ А.

Исходный код программы. Файл 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 plt


def read_data():
    dataframe = pandas.read_csv("sample_data/sonar.csv", header=None)
    dataset = dataframe.values
    x = dataset[:, 0:60].astype(float)
    y = dataset[:, 60]
    return x, y


def encode_y(raw_y):
    encoder = LabelEncoder()
    encoder.fit(raw_y)
    encoded_y = encoder.transform(raw_y)
    y_mapping = dict(zip(encoder.classes_,
encoder.transform(encoder.classes_)))
    return encoded_y, y_mapping


def plot(epochs, train, validation, metrics):
    plt.plot(epochs, train, 'b', label=f'Training {metrics}')
    plt.plot(epochs, validation, 'r', label=f'Validation {metrics}')
    plt.title(f'Training and validation {metrics}')
    plt.xlabel('Epochs')
    plt.ylabel(metrics.capitalize())
    plt.grid(True)
    plt.legend()


def plot_history(history):
    loss = history['loss']
    val_loss = history['val_loss']
    acc = history['accuracy']
    val_acc = history['val_accuracy']
    epochs = range(1, len(loss) + 1)

    plt.figure()
    plt.subplot(211)
    plot(epochs, loss, val_loss, "loss")
    plt.subplot(212)
    plot(epochs, acc, val_acc, "accuracy")
    plt.show()


def build_model():
    model = Sequential()
```

```
    model.add(Dense(60, input_dim=60, kernel_initializer='normal',
activation='relu'))
    #model.add(Dense(30, input_dim=30, kernel_initializer='normal',
activation='relu'))
    #model.add(Dense(30, kernel_initializer='normal', activation='relu'))
    #model.add(Dense(15, kernel_initializer='normal', activation='relu'))
    model.add(Dense(1, kernel_initializer='normal',
activation='sigmoid'))

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


X, Y = read_data()
#X = X[:, :30]
Y, mapping = encode_y(Y)
model_ = build_model()
h = model_.fit(X, Y, epochs=100, batch_size=10, validation_split=0.1)
plot_history(h.history)
```