1. Эксперимент 1-й

В ходе первого эксперимента, было решено использовать следующую архитектуру:

Обучение:

model = Sequential()

model.add(Conv2D(32, (3, 3), input\_shape=input\_shape))

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Conv2D(32, (3, 3)))

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Flatten())

model.add(Dense(64))

model.add(Activation('relu'))

model.add(Dropout(0.4))

model.add(Dense(1))

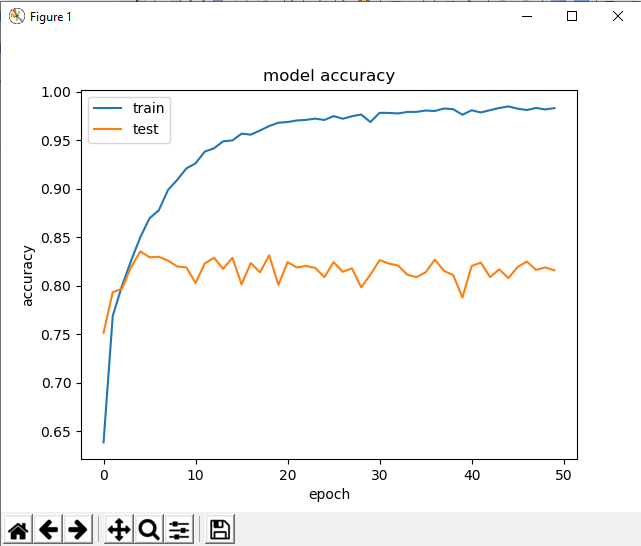
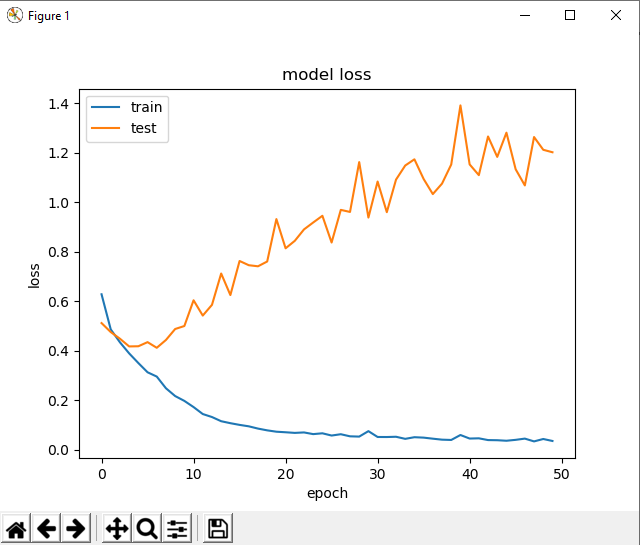
model.add(Activation('sigmoid'))

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

history = model.fit\_generator(train\_gen, steps\_per\_epoch=200, epochs=50, validation\_data = val\_gen, validation\_steps=100)

Результат:

77.4995512919331 – процентов точности при оценивании модели.



1. Эксперимент 2-й

model = Sequential()

model.add(Conv2D(32, (3, 3), input\_shape=input\_shape))

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Conv2D(32, (3, 3)))

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Flatten())

model.add(Dense(500))

model.add(Activation('relu'))

model.add(Dropout(0.5))

model.add(Dense(64))

model.add(Activation('relu'))

model.add(Dropout(0.5))

model.add(Dense(2))

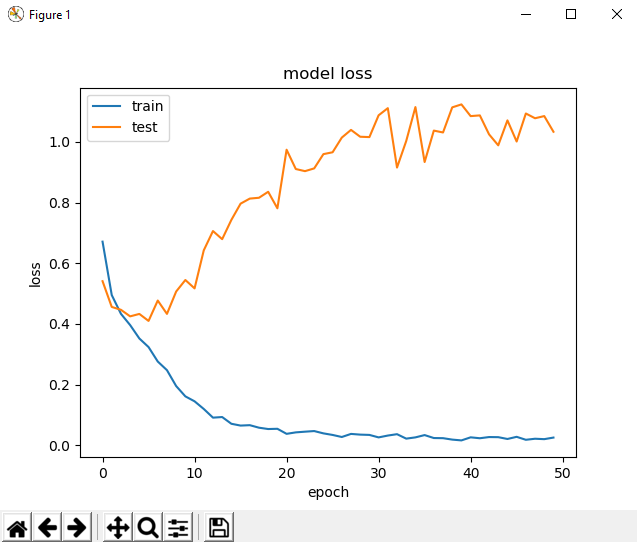
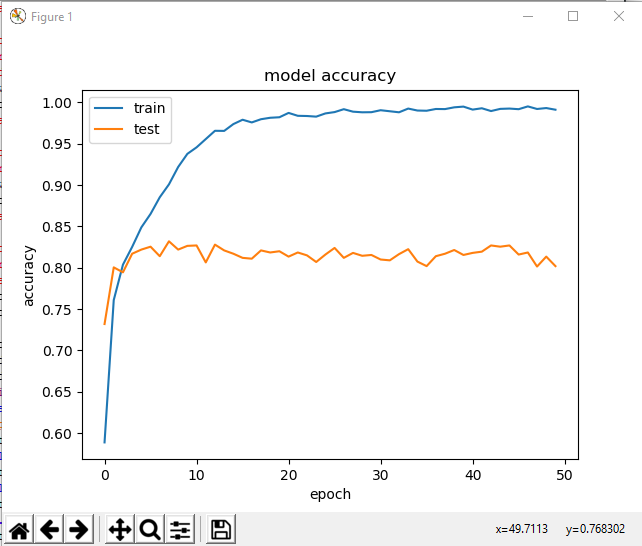
model.add(Activation('softmax'))

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

history = model.fit\_generator(train\_gen, steps\_per\_epoch=200, epochs=50, validation\_data = val\_gen, validation\_steps=100)

Результат:

0.7838359363490772 – процентов точности.



1. Эксперимент 3-й

model = Sequential()

model.add(Conv2D(32, (3, 3), input\_shape=input\_shape))

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Conv2D(32, (3, 3)))

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Flatten())

model.add(Dense(80))

model.add(Activation('relu'))

model.add(Dropout(0.5))

model.add(Dense(80))

model.add(Activation('relu'))

model.add(Dropout(0.5))

model.add(Dense(2))

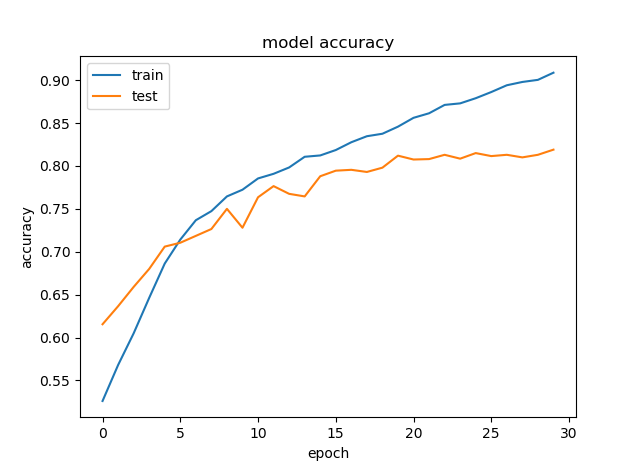
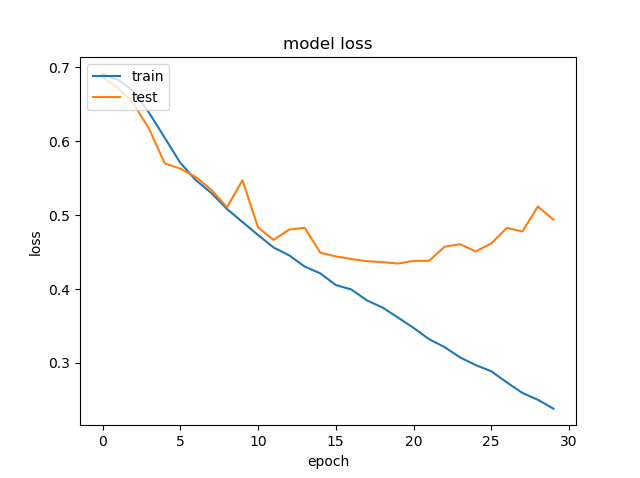
model.add(Activation('softmax'))

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

history = model.fit\_generator(train\_gen, steps\_per\_epoch=200, epochs=30, validation\_data = val\_gen, validation\_steps=100)

Результат:

0.7800215406962216 – процентов точности.



1. Эксперимент 4-й

model = Sequential()

model.add(Conv2D(32, (3, 3), input\_shape=input\_shape))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Conv2D(128, (3, 3)))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Flatten())

model.add(Dense(80))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(Dropout(0.5))

model.add(Dense(80))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(Dropout(0.5))

model.add(Dense(2))

model.add(Activation('softmax'))

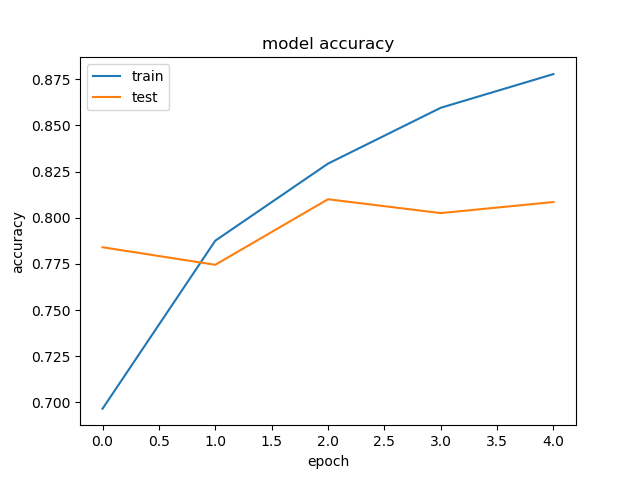
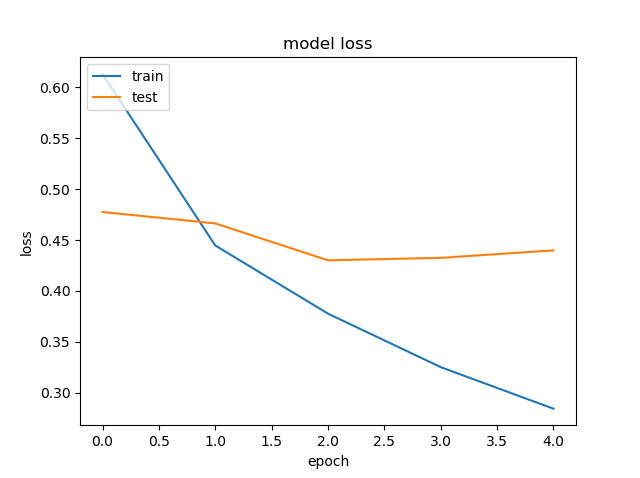
model.compile(loss = 'categorical\_crossentropy', optimizer = 'SGD', metrics=['accuracy'])

history = model.fit\_generator(train\_gen, steps\_per\_epoch=200, epochs=30, validation\_data = val\_gen, validation\_steps=100, callbacks=[esc])

В этом эксперименте, количество слов возросло по сравнению с предыдущими моделями, время на обучение стало уходить гораздо больше и следствие из этого было применено: keras.callbacks «EarlyStopping». Ранняя остановка обучения при условии, что точность на проверочном наборе не больше, чем на обучающей выборке, в течении 2-х эпох.

Точность:

0.7959073780094822 – процентов точности.



1. Эксперимент 5-й

model = Sequential()

model.add(Conv2D(32, (3, 3), input\_shape=input\_shape))

model.add(Activation('relu'))

model.add(Conv2D(32, (3, 3), input\_shape=input\_shape))

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.1))

model.add(Conv2D(64, (3, 3)))

model.add(Activation('relu'))

model.add(Conv2D(64, (3, 3)))

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.1))

model.add(Flatten())

model.add(Dense(200))

model.add(Activation('relu'))

model.add(Dropout(0.5))

model.add(Dense(2))

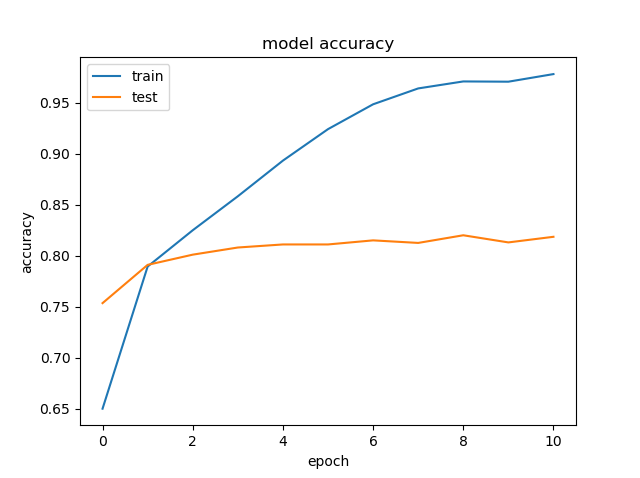
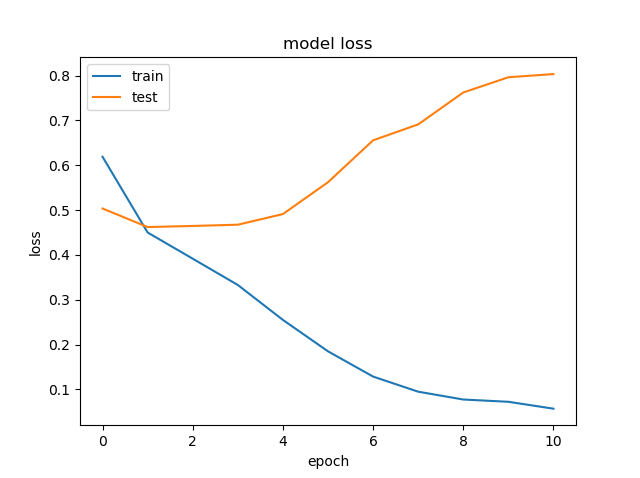
model.add(Activation('softmax'))

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

history = model.fit\_generator(train\_gen, steps\_per\_epoch=200, epochs=30, validation\_data = val\_gen, validation\_steps=100, callbacks=[esc])

Точность:

0.7769700234956147 – процентов точности.



1. Эксперимент 6-й

model = Sequential()

model.add(Conv2D(32, (3, 3), input\_shape=input\_shape))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(Conv2D(32, (3, 3), input\_shape=input\_shape))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.2))

model.add(Conv2D(64, (3, 3)))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(Conv2D(64, (3, 3)))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.2))

model.add(Conv2D(128, (3, 3)))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(Conv2D(128, (3, 3)))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.2))

model.add(Flatten())

model.add(Dense(80))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(Dropout(0.55))

model.add(Dense(80))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(Dropout(0.55))

model.add(Dense(2))

model.add(Activation('softmax'))

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

history = model.fit\_generator(train\_gen, steps\_per\_epoch=200, epochs=30, validation\_data = val\_gen, validation\_steps=100, callbacks=[esc])

Точность:

0.800753904050181 – процентов точности.