Лабораторная работа 2. Сверточные сети.

**Выполнил:** Кириллов Данил  
**Курс:** 4  
**Группа:** ФЗ-11

# Эксперимент 0 – стартовый.

|  |
| --- |
| model = Sequential()  model.add(Conv2D(16, (3,3), input\_shape=(32,32,3), padding='same'))  model.add(BatchNormalization())  model.add(ReLU())  model.add(MaxPooling2D(2,2))  *for* i in range(50):      model.add(Conv2D(64,(3,3), padding='same'))      model.add(BatchNormalization())      model.add(ReLU())  model.add(MaxPooling2D(2,2))  model.add(Flatten())  model.add(Dense(num\_classes))  model.add(Activation('softmax'))  opt = keras.optimizers.Adam(0.001)  model.compile(loss='categorical\_crossentropy',                optimizer=opt,                metrics=['accuracy'])  model.summary() |
|  |

# Эксперимент 1. Уменьшение сверточных слоев.

|  |
| --- |
| model = Sequential()  model.add(Conv2D(16, (3,3), input\_shape=(32,32,3), padding='same'))  model.add(BatchNormalization())  model.add(ReLU())  model.add(MaxPooling2D(2,2))  *for* i in range(25):      model.add(Conv2D(64,(3,3), padding='same'))      model.add(BatchNormalization())      model.add(ReLU())  model.add(MaxPooling2D(2,2))  model.add(Flatten())  model.add(Dense(num\_classes))  model.add(Activation('softmax'))  opt = keras.optimizers.Adam(0.001)  model.compile(loss='categorical\_crossentropy',                optimizer=opt,                metrics=['accuracy'])  model.summary() |
|  |

При уменьшении количества сверточных слоев в 2 раза незначительно увеличилась точность (+3 %), скорость обучение уменьшилась в 2 раза.

# Эксперимент 2. Уменьшение сверточных слоев + увеличение количества фильтров

|  |
| --- |
| model = Sequential()  model.add(Conv2D(16, (3,3), input\_shape=(32,32,3), padding='same'))  model.add(BatchNormalization())  model.add(ReLU())  model.add(MaxPooling2D(2,2))  num\_filters = 16  *for* i in range(4):      num\_filters \*= 2      model.add(Conv2D(num\_filters, (3,3), padding='same'))      model.add(BatchNormalization())      model.add(ReLU())  model.add(MaxPooling2D(2,2))  model.add(Flatten())  model.add(Dense(num\_classes))  model.add(Activation('softmax'))  opt = keras.optimizers.Adam(0.001)  model.compile(loss='categorical\_crossentropy',                optimizer=opt,                metrics=['accuracy'])  model.summary() |
|  |

Вместо константного значения количества фильтров, использовалось последовательное удваивание их на каждом слое. Также количество слоев уменьшилось в 6 раз, по сравнению с предыдущим экспериментом, а время на обучение уменьшилось в 1.5 раза

# Эксперимент 3. Изменение оптимизатора.

|  |
| --- |
| model = Sequential()  model.add(Conv2D(16, (3,3), input\_shape=(32,32,3), padding='same'))  model.add(BatchNormalization())  model.add(ReLU())  model.add(MaxPooling2D(2,2))  num\_filters = 16  *for* i in range(4):      num\_filters \*= 2      model.add(Conv2D(num\_filters, (3,3), padding='same'))      model.add(BatchNormalization())      model.add(ReLU())  model.add(MaxPooling2D(2,2))  model.add(Flatten())  model.add(Dense(num\_classes))  model.add(Activation('softmax'))  opt = keras.optimizers.RMSprop(0.001)  model.compile(loss='categorical\_crossentropy',                optimizer=opt,                metrics=['accuracy'])  model.summary() |
|  |

Изменение оптимизатора на RMSprop ухудшило точность модели, а также обучение проходило менее стабильно.

# Эксперимент 4. Изменение оптимизатора.

|  |
| --- |
| model = Sequential()  model.add(Conv2D(16, (3,3), input\_shape=(32,32,3), padding='same'))  model.add(BatchNormalization())  model.add(ReLU())  model.add(MaxPooling2D(2,2))  num\_filters = 16  *for* i in range(4):      num\_filters \*= 2      model.add(Conv2D(num\_filters, (3,3), padding='same'))      model.add(BatchNormalization())      model.add(ReLU())  model.add(MaxPooling2D(2,2))  model.add(Flatten())  model.add(Dense(num\_classes))  model.add(Activation('softmax'))  opt = keras.optimizers.Nadam(0.001)  model.compile(loss='categorical\_crossentropy',                optimizer=opt,                metrics=['accuracy'])  model.summary() |
|  |

Изменение оптимизатора на Nadam(0.001) также не привело к улучшениям результатов.

# Эксперимент 5. Изменение batch size.

|  |
| --- |
| model = Sequential()  model.add(Conv2D(16, (3,3), input\_shape=(32,32,3), padding='same'))  model.add(BatchNormalization())  model.add(ReLU())  model.add(MaxPooling2D(2,2))  num\_filters = 16  *for* i in range(4):      num\_filters \*= 2      model.add(Conv2D(num\_filters, (3,3), padding='same'))      model.add(BatchNormalization())      model.add(ReLU())  model.add(MaxPooling2D(2,2))  model.add(Flatten())  model.add(Dense(num\_classes))  model.add(Activation('softmax'))  opt = keras.optimizers.Adam(0.001)  model.compile(loss='categorical\_crossentropy',                optimizer=opt,                metrics=['accuracy'])  model.summary() |
|  |

Увеличение размера batch также не привело к улучшениям результатов.

# Эксперимент 6. Увеличение сверточных слоев.

|  |
| --- |
| model = Sequential()  model.add(Conv2D(16, (3,3), input\_shape=(32,32,3), padding='same'))  model.add(BatchNormalization())  model.add(ReLU())  model.add(MaxPooling2D(2,2))  num\_filters = 16  max\_filter = 128  *for* i in range(10):  *if* num\_filters < max\_filter:          num\_filters \*= 2      model.add(Conv2D(num\_filters, (3,3), padding='same'))      model.add(BatchNormalization())      model.add(ReLU())  model.add(MaxPooling2D(2,2))  model.add(Flatten())  model.add(Dense(num\_classes))  model.add(Activation('softmax'))  opt = keras.optimizers.Adam(0.001)  model.compile(loss='categorical\_crossentropy',                optimizer=opt,                metrics=['accuracy'])  model.summary() |
|  |

Увеличиваем количество сверточных слоев до 10, с постепенным увеличением количества фильтров до 128. Также уменьшаем batch size до 80.

В результате увеличиваем точность модели на 1%.

# Эксперимент 7. Увеличение эпох и количества сверточных слоев с ограничение фильтров.

|  |
| --- |
| model = Sequential()  model.add(Conv2D(16, (3,3), input\_shape=(32,32,3), padding='same'))  model.add(BatchNormalization())  model.add(ReLU())  model.add(MaxPooling2D(2,2))  num\_filters = 16  max\_filter = 128  *for* i in range(6):  *if* num\_filters < max\_filter:          num\_filters \*= 2      model.add(Conv2D(num\_filters, (3,3), padding='same'))      model.add(BatchNormalization())      model.add(ReLU())  model.add(MaxPooling2D(2,2))  model.add(Flatten())  model.add(Dense(num\_classes))  model.add(Activation('softmax'))  opt = keras.optimizers.Adam(0.001)  model.compile(loss='categorical\_crossentropy',                optimizer=opt,                metrics=['accuracy'])  model.summary() |
|  |

# Сводная таблица.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Кол-во эпох | Оптимизатор | Размер batch | v\_loss | v\_acc | Время |
| 0 | 30 | Adam(0.001) | 64 | 0.81 | 74% ( - ) | 30 мин |
| 1 | 30 | Adam(0.001) | 64 | 0.77 | 77% (**+3%**) | 15 мин |
| 2 | 30 | Adam(0.001) | 64 | 0.56 | 84% (**+7%**) | 10 мин |
| 3 | 30 | RMSprop(0.001) | 64 | 0.59 | 81% (**-3%**) | 15 мин |
| 4 | 30 | Nadam(0.001) | 64 | 0.57 | 82% (**+1%**) | 15 мин |
| 5 | 30 | Adam(0.001) | 128 | 0.53 | 83%(**+1%**) | 10 мин |
| 6 | 30 | Adam(0.001) | 80 | 0.48 | 85%(**+1%**) | 11 мин |
| 7 | 50 | Adam(0.001) | 80 | 0.5 | 86%(**+1%**) | 18 мин |

# Итоги.

В ходе выполнения лабораторной работы удалось получить модель, которая классифицирует изображения с точностью валидации (v\_acc) на уровне 86%.