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

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Искусственные нейронные сети»

Тема: Распознавание объектов на фотографиях

Студент гр. 8383 _____ Ларин А.

Преподаватель _____ Жангиров Т.Р.

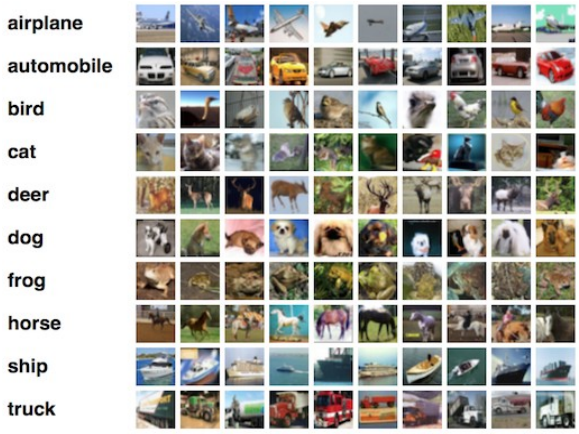
Санкт-Петербург

2021

Цель работы

Распознавание объектов на фотографиях (Object Recognition in Photographs)

CIFAR-10 (классификация небольших изображений по десяти классам: самолет, автомобиль, птица, кошка, олень, собака, лягушка, лошадь, корабль и грузовик).



Задание

- Ознакомиться со сверточными нейронными сетями
- Изучить построение модели в Keras в функциональном виде
- Изучить работу слоя разреживания (Dropout)

Требования

1. Построить и обучить сверточную нейронную сеть
2. Исследовать работу сеть без слоя Dropout
3. Исследовать работу сети при разных размерах ядра свертки

Выполнение

Был загружен набор данных CIFAR-10 из Keras
Значения были нормализованы, а классы преобразованы к категориальным

Листинг 1:

```
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= np.max(X_train) # Normalise data to [0, 1] range
X_test /= np.max(X_train) # Normalise data to [0, 1] range
Y_train = utils.to_categorical(y_train, num_classes) # One-hot encode the labels
Y_test = utils.to_categorical(y_test, num_classes) # One-hot encode the labels
```

Далее была создана сверточная сеть для классификации.

Сеть состоит из четырех слоев Convolution_2D и слоев MaxPooling2D после второй и четвертой сверток. После первого слоя подвыборки удваивается количество ядер. После этого выходное изображение слоя подвыборки трансформируется в одномерный вектор (слоем Flatten) и проходит два полносвязных слоя (Dense). На всех слоях, кроме выходного полносвязного слоя, используется функция активации ReLU. Последний слой использует softmax.

Для регуляризации модели после каждого слоя подвыборки и первого полносвязного слоя применяется слой Dropout.
Модель была построена в функциональном виде

Листинг 2:
```
inp = Input(shape=(depth, height, width)) # N.B. depth goes first in Keras
# Conv [32] -> Conv [32] -> Pool (with dropout on the pooling layer)
conv_1 = Convolution2D(conv_depth_1, (kernel_size, kernel_size), padding='same',
activation='relu')(inp)
conv_2 = Convolution2D(conv_depth_1, (kernel_size, kernel_size), padding='same',
activation='relu')(conv_1)
pool_1 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_2)
drop_1 = Dropout(drop_prob_1)(pool_1)
# Conv [64] -> Conv [64] -> Pool (with dropout on the pooling layer)
conv_3 = Convolution2D(conv_depth_2, (kernel_size, kernel_size), padding='same',
activation='relu')(drop_1)
conv_4 = Convolution2D(conv_depth_2, (kernel_size, kernel_size), padding='same',
activation='relu')(conv_3)
pool_2 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_4)
drop_2 = Dropout(drop_prob_1)(pool_2)
# Now flatten to 1D, apply Dense -> ReLU (with dropout) -> softmax
flat = Flatten()(drop_2)
hidden = Dense(hidden_size, activation='relu')(flat)
drop_3 = Dropout(drop_prob_2)(hidden)
out = Dense(num_classes, activation='softmax')(drop_3)

model = Model(inputs=inp, outputs=out)
```

Модель скомпилирована с кроссэнтропией в качестве потерь, оптимизатором Адам и метрикой точность

Затем модель обучена на 20-ти эпохах cbatch size = 32 и оценена

Листинг 3:
```
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
hist = model.fit(X_train, Y_train,
          batch_size=batch_size, epochs=num_epochs,
          verbose=1, validation_split=0.1)
model.evaluate(X_test, Y_test, verbose=1)
```

Результат на рис. 1
Точность при обучении и валидации составила 0.8631 и 0.7918 соответственно

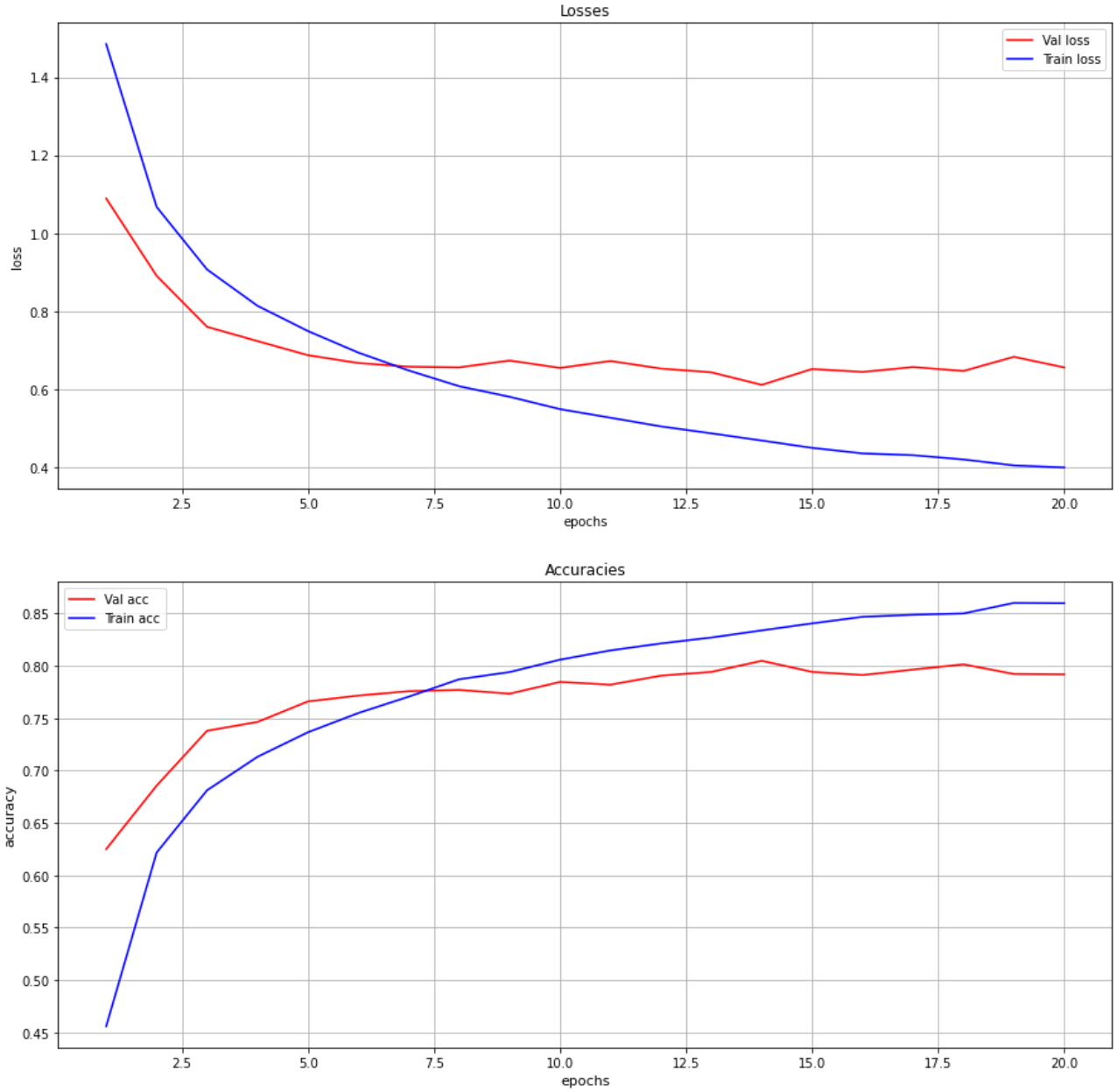


Рисунок 1 — Первый результат обучения модели

Затем из сети были удалены слои Dropout и обучение проведено заново.

Результат на рис. 2
очевидно переобучение — после четырех эпох точность при валидации начала падать, а потери быстро возрастать.
Точность при обучении и валидации составила 0.9850 и 0.7488 соответственно

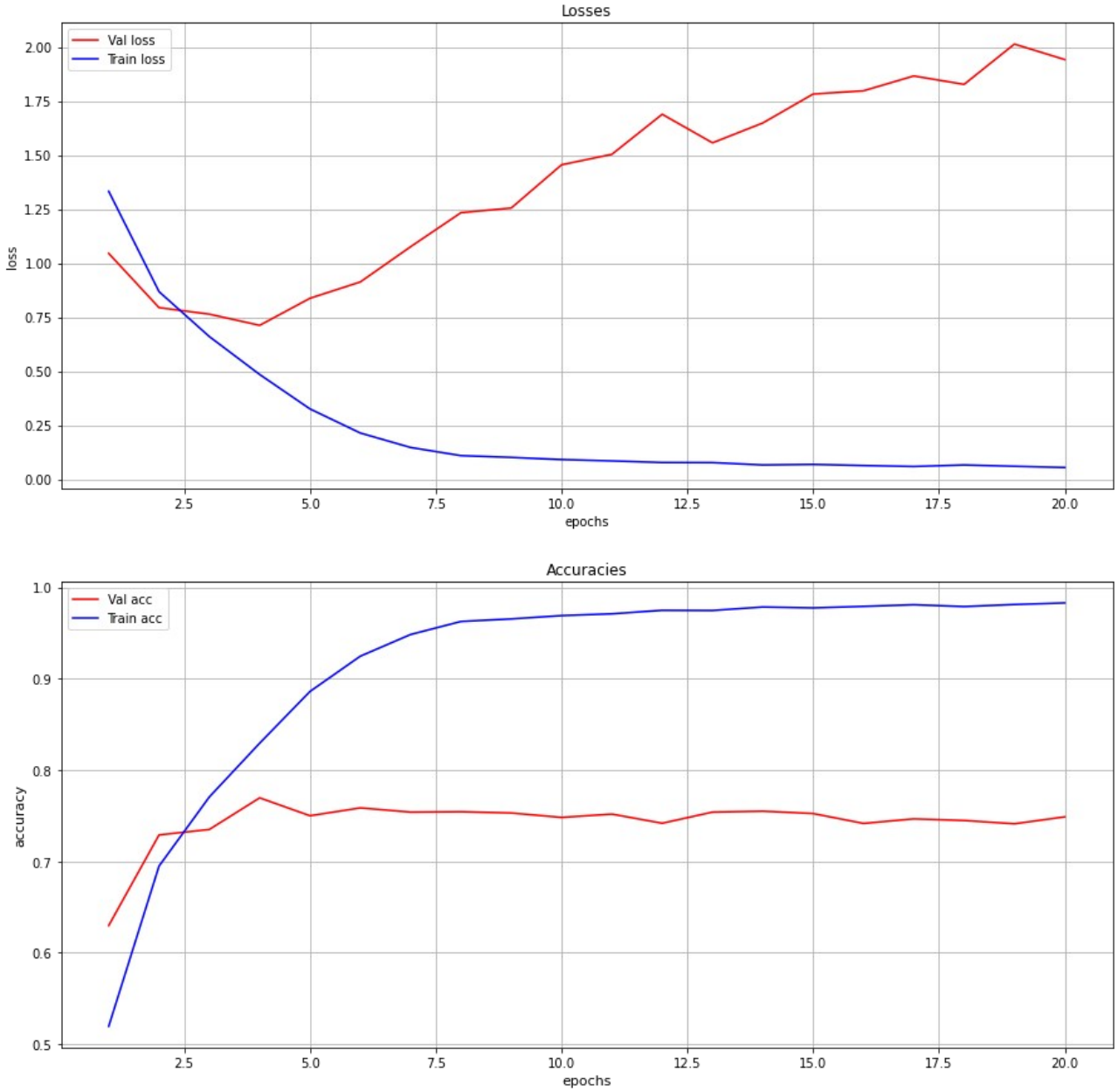


Рисунок 2 — Результат обучения модели без dropout

Dropout был включен обратно в модель
Проведен эксперимент с варьированием размера ядра. В предыдущих экспериментах он был равен 3*3. Установлен размер ядра 2
График обучения на рис. 3

Точность при обучении и валидации составила 0.8407 и 0.7804 соответственно
Точность немного упала по сравнению с первым экспериментом.

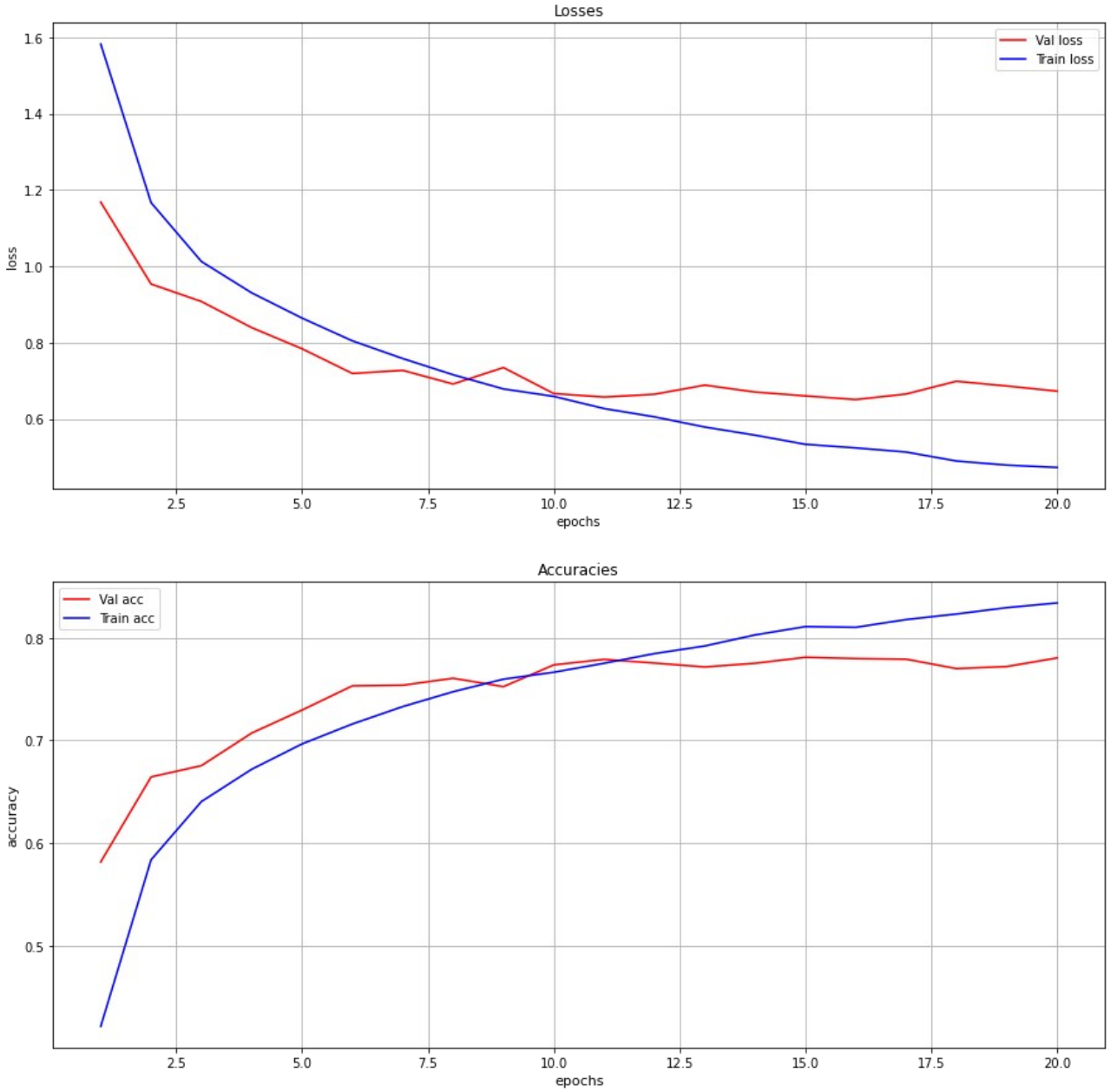


Рисунок 3 — Результат обучения модели с ядром 2*2

Далее размер ядра увеличен до 5

График на рис.4
Точность при обучении и валидации составила 0.8095 и 0.7712 соответственно

Точность упала по сравнению с первым и третьим экспериментом

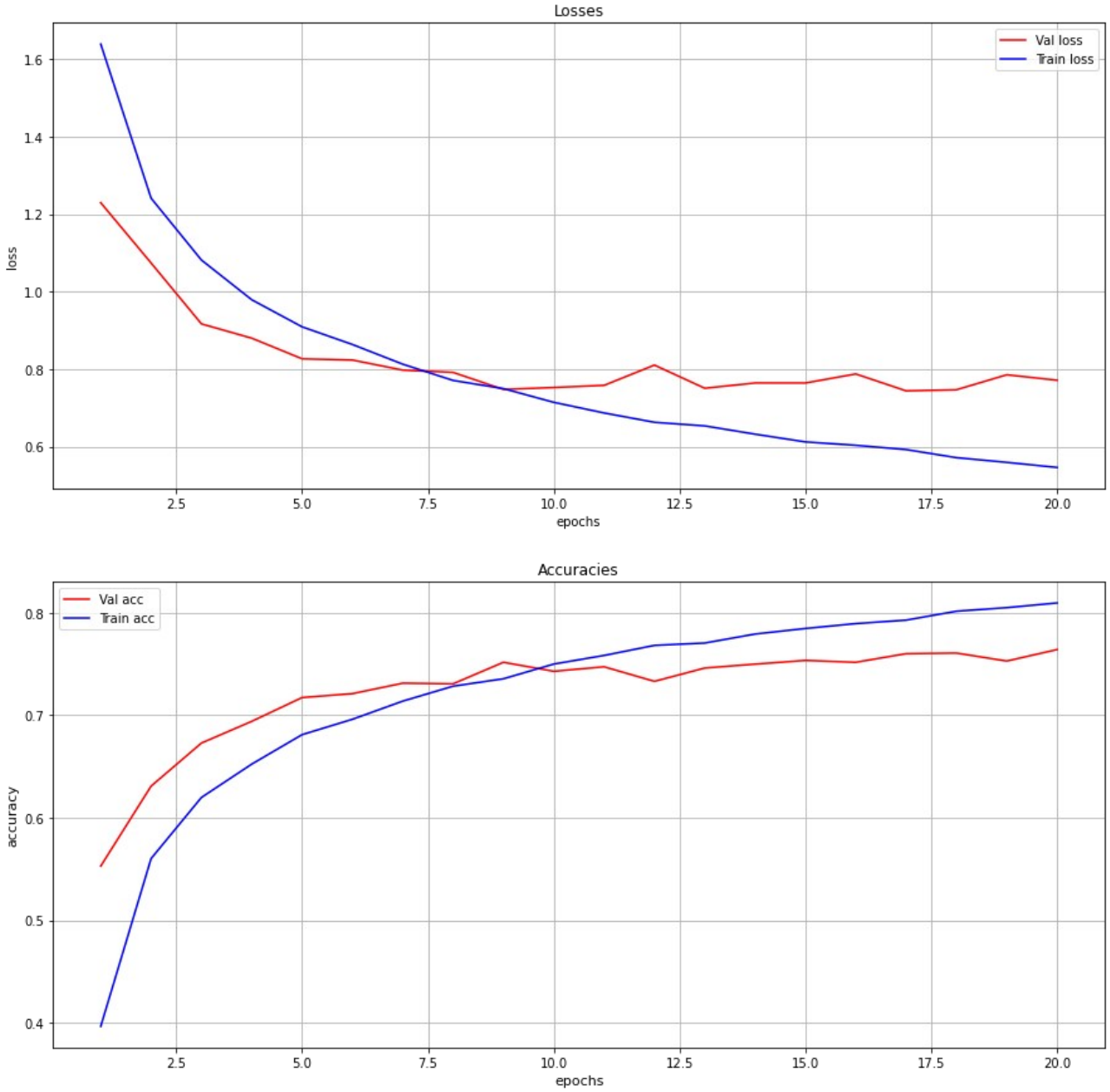


Рисунок 4 — Результат обучения модели с ядром 5*5

Размер ядра установлен 4*4
Точность при обучении и валидации составила 0.8153 и 0.7692 соответственно, что хуже чем при ядре 3*3

Выводы.

Была обучена CNN модель для классификации изображений.
Были проведены с включением и выключением слоев dropout. При выключении dropout очность при обучении возросла, однако произошло переобучение, и точность при валидации понизилась.

Были проведены эксперименты с различными размерами ядра. Лучшие результаты поулчена для язда 3*3, при больших или меньших размерах точность падает.