Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Кафедра информатики

Отчет по лабораторной работе #5

«**Применение сверточных нейронных сетей (бинарная классификация)**»

Карп Александр Игоревич

магистрант кафедры информатики

группа №858641

Минск 2019

**Задание 1.**

Загрузите данные. Разделите исходный набор данных на обучающую, валидационную и контрольную выборки.

PATH = "C:\\Users\\karps\\Downloads\\dogs-vs-cats\\train\\train"  
  
IMAGE\_WIDTH=128  
IMAGE\_HEIGHT=128  
IMAGE\_SIZE=(IMAGE\_WIDTH, IMAGE\_HEIGHT)  
IMAGE\_CHANNELS=3  
filenames = os.listdir(PATH)  
categories = []  
for filename in filenames:  
 category = filename.split('.')[0]  
 if category == 'dog':  
 categories.append(1)  
 else:  
 categories.append(0)  
  
df = pd.DataFrame({  
 'filename': filenames,  
 'category': categories  
})  
  
df["category"] = df["category"].replace({0: 'cat', 1: 'dog'})  
train\_df, validate\_df = train\_test\_split(df, test\_size=0.20, random\_state=42)  
train\_df = train\_df.reset\_index(drop=True)  
validate\_df = validate\_df.reset\_index(drop=True)

**Задание 2.**

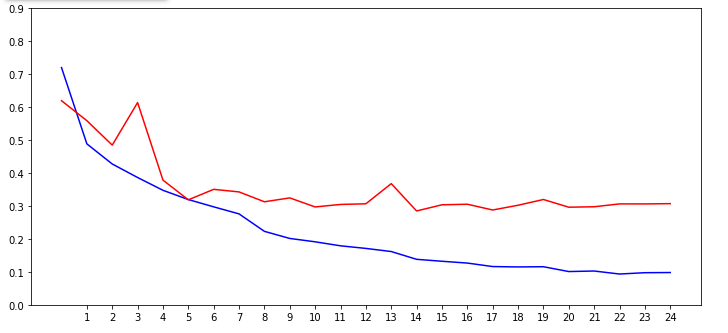
Реализуйте глубокую нейронную сеть с как минимум тремя сверточными слоями. Какое качество классификации получено?

model = Sequential()  
model.add(Conv2D(32, (3, 3), activation='relu', input\_shape=(128, 128, 1)))  
model.add(BatchNormalization())  
model.add(MaxPooling2D(pool\_size=(2, 2)))  
model.add(Dropout(0.25))  
  
model.add(Conv2D(64, (3, 3), activation='relu'))  
model.add(BatchNormalization())  
model.add(MaxPooling2D(pool\_size=(2, 2)))  
model.add(Dropout(0.25))  
  
model.add(Conv2D(128, (3, 3), activation='relu'))  
model.add(BatchNormalization())  
model.add(MaxPooling2D(pool\_size=(2, 2)))  
model.add(Dropout(0.25))  
  
model.add(Flatten())  
model.add(Dense(512, activation='relu'))  
model.add(BatchNormalization())  
model.add(Dropout(0.5))  
model.add(Dense(2, activation='softmax')) # 2 because we have cat and dog classes  
  
model.compile(loss='categorical\_crossentropy', optimizer='rmsprop', metrics=['accuracy'])

datagen = ImageDataGenerator(  
 rescale=1./255,)  
  
  
total\_train = train\_df.shape[0]  
total\_validate = validate\_df.shape[0]  
batch\_size=15  
train\_generator = datagen.flow\_from\_dataframe(  
 train\_df,  
 PATH,  
 x\_col='filename',  
 y\_col='category',  
 target\_size=IMAGE\_SIZE,  
 class\_mode='categorical',  
 batch\_size=batch\_size  
)  
validation\_datagen = ImageDataGenerator(rescale=1./255)  
validation\_generator = validation\_datagen.flow\_from\_dataframe(  
 validate\_df,  
 PATH,  
 x\_col='filename',  
 y\_col='category',  
 target\_size=IMAGE\_SIZE,  
 class\_mode='categorical',  
 batch\_size=batch\_size  
)

history = model.fit\_generator(  
 train\_generator,  
 epochs=50,  
 validation\_data=validation\_generator,  
 validation\_steps=total\_validate//batch\_size,  
 steps\_per\_epoch=total\_train//batch\_size,  
 callbacks=callbacks  
)

В данном случае генератор данных делает одно действие – нормализует изображение (вместо значений 0-255 пиксели становятся 0-1)



**Задание 3.**

Примените дополнение данных (data augmentation). Как это повлияло на качество классификатора?

Изменим лишь генератор данной сети и заново ее обучим

datagen = ImageDataGenerator(

    rotation\_range=15,

    rescale=1./255,

    shear\_range=0.1,

    zoom\_range=0.2,

    horizontal\_flip=True,

    width\_shift\_range=0.1,

    height\_shift\_range=0.1

)

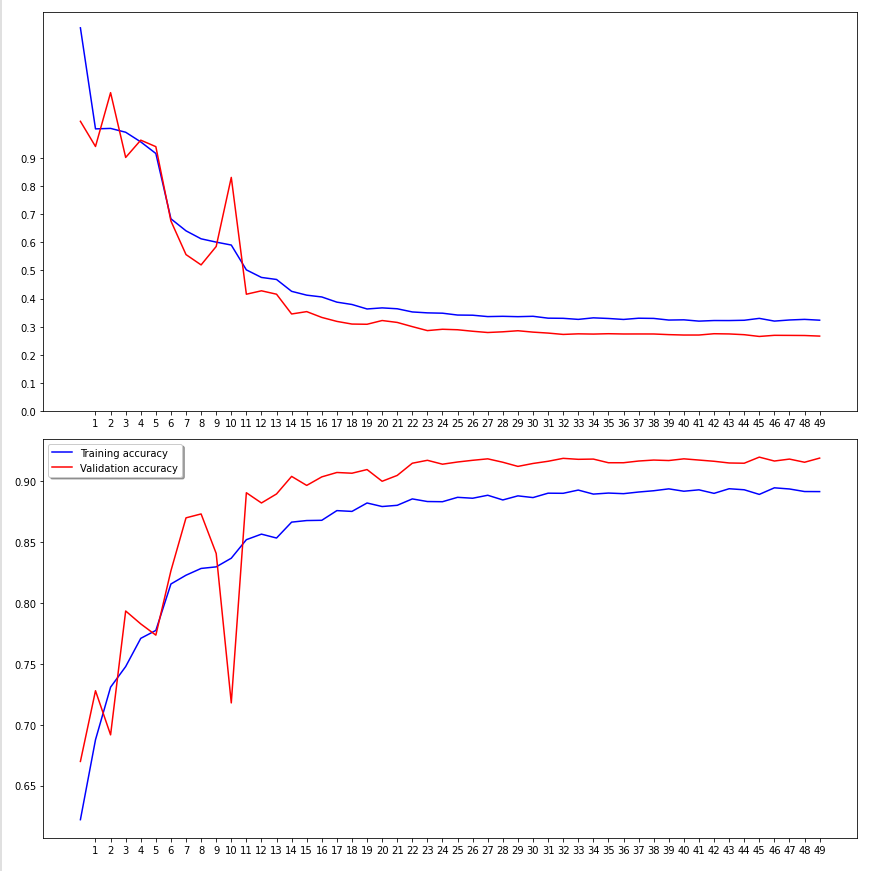


Рис 2. Графики зависимости функции потерь и точность классификации от эпохи обучения

Благодаря data augmentation удалось улучшить классификацию с 0.901 до 0.919, почти на 2 процента.

base\_model=VGG16(weights='imagenet',include\_top=False, input\_shape=(128,128,3)) #imports the mobilenet model and discards the last 1000 neuron layer.  
  
x=base\_model.output  
x=GlobalAveragePooling2D()(x)  
x=Dense(1024,activation='relu',kernel\_regularizer=regularizers.l2(0.01))(x) #we add dense layers so that the model can learn more complex functions and classify for better results.  
x = Dropout(0.2)(x)  
x=Dense(1024,activation='relu', kernel\_regularizer=regularizers.l2(0.01))(x) #dense layer 2  
x = Dropout(0.2)(x)  
x=Dense(512,activation='relu', kernel\_regularizer=regularizers.l2(0.01))(x) #dense layer 3  
x = Dropout(0.2)(x)  
preds=Dense(2,activation='softmax')(x) #final layer with softmax activationcallbacks=callbacks  
model=Model(inputs=base\_model.input,outputs=preds)  
  
for layer in model.layers[:20]:  
 layer.trainable=False  
for layer in model.layers[20:]:  
 layer.trainable=True  
  
  
model.compile(loss='categorical\_crossentropy', optimizer='rmsprop', metrics=['accuracy'])  
history = model.fit\_generator(  
 train\_generator,  
 epochs=50,  
 validation\_data=validation\_generator,  
 validation\_steps=total\_validate//batch\_size,  
 steps\_per\_epoch=total\_train//batch\_size,

Воспользуемся преобученной сетью VGG16 из библиотеки keras, добавив несколько полносвязных слоев в конце.

Сеть показала высокий результат с первой эпохи, дальше можно заметить результат переобучения – точность на тренировочном сете растет, а на валидационном – нет. Максимальная достигнутая точность составила 0.88.