# Домашнее задание. Нейронные сети

**Вопрос 1**

Так как перед нами стоит задача бинарной классификации, какую функцию потерь лучше всего применить в нашем случае?

- **бинарная кросс-энтропия (binary crossentropy)**
- фокусные потери (focal loss)
- среднеквадратичная ошибка (mean squared error)
- категориальная кросс-энтропия (categorical crossentropy)

**Примечание**: Если вы используете активацию для выходного слоя, не требуется устанавливать `from_logits=True`.

**Вопрос 2**

Определите общее количество параметров в модели. Для этого примените метод `summary`.

- 9215873
- **11215873**
- 14215873
- 19215873

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 148, 148, 32)      896

 max_pooling2d_2 (MaxPoolin  (None, 74, 74, 32)        0
 g2D)

 flatten_2 (Flatten)         (None, 175232)            0

 dense_4 (Dense)             (None, 64)                11214912

 dense_5 (Dense)             (None, 1)                 65

=================================================================
Total params: 11215873 (42.79 MB)
Trainable params: 11215873 (42.79 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

**Генераторы и обучение**

Для следующих двух вопросов используйте следующий генератор данных для обучающих (train) и тестовых (test) наборов:

`ImageDataGenerator(rescale=1./255)`

**Примечание**: Дополнительная предобработка изображений не требуется. При загрузке данных из каталогов обучения/тестирования убедитесь, что параметр `class_mode` установлен правильно для задачи бинарной классификации. Рекомендуемые параметры: `batch_size=20` и `shuffle=True`.

Для обучения примените метод `.fit()` со следующими параметрами:

`model.fit( train_generator, epochs=10, validation_data=test_generator)`

**Вопрос 3**

Какова медиана точности обучения по всем эпохам?

- 0.40
- 0.60 (0,708 – мой ответ)
- 0.90
- 0.20
-

```
# Получаем значения точности обучения для всех эпох
train_accuracy = history.history['accuracy']

# Вычисляем медиану точности обучения
median_train_accuracy = np.median(train_accuracy)

print("Медиана точности обучения по всем эпохам:", median_train_accuracy)
```

```
Медиана точности обучения по всем эпохам: 0.7081218361854553
```

**Вопрос 4**

Каково стандартное отклонение потерь в процессе обучения по всем эпохам?
Не подходит, у меня

```
Стандартное отклонение потерь в процессе обучения по всем эпохам:
0.061030830225842686
```

- 0.11
- 0.66
- 0.99
- 0.33

```
# Получаем значения потерь обучения для всех эпох
train_loss = history.history['loss']

# Вычисляем стандартное отклонение потерь обучения
std_train_loss = np.std(train_loss)

print("Стандартное отклонение потерь в процессе обучения по всем эпохам:", std_train_loss)
# Оцениваем модель
loss, accuracy = model.evaluate(test_generator)
print(f'Потери на тесте: {loss}, Точность на тесте: {accuracy}')
```

```
Стандартное отклонение потерь в процессе обучения по всем эпохам: 0.061030830225842686
80/80 [==============================] - 19s 239ms/step - loss: 0.6879 - accuracy: 0.5577
Потери на тесте: 0.6878818273544312, Точность на тесте: 0.557716429233551
```

- 

**Аугментация данных**

Для следующего этапа вам потребуется генерировать больше данных с помощью аугментаций.

Добавьте следующие аугментации к генератору обучающих данных:

rotation_range=40,
width_shift_range=0.2,
height_shift_range=0.2,
shear_range=0.2,
zoom_range=0.2,
horizontal_flip=True,
fill_mode='nearest'

**Вопрос 5**

Обучите модель еще на 10 эпох с использованием указанного выше кода. Не создавайте модель с нуля; продолжите обучение существующей.

Каково среднее значение потерь на тестовом наборе данных по всем эпохам после аугментации?

- 0.15
- 0.77  (мой ответ 0,68)
- 0.37
- 0.97

```
#
# Продолжаем обучение модели еще на 10 эпох
history_additional = model.fit(
    train_generator,
    epochs=10,
    validation_data=test_generator
)
```

```
Epoch 1/10
20/20 [==============================] - 33s 2s/step - loss: 0.7144 - accuracy: 0.4975 - val_loss: 0.6901 - val_accuracy: 0.5188
Epoch 2/10
20/20 [==============================] - 33s 2s/step - loss: 0.6935 - accuracy: 0.4822 - val_loss: 0.6926 - val_accuracy: 0.4818
Epoch 3/10
20/20 [==============================] - 33s 2s/step - loss: 0.6915 - accuracy: 0.5102 - val_loss: 0.6880 - val_accuracy: 0.5213
Epoch 4/10
20/20 [==============================] - 33s 2s/step - loss: 0.6924 - accuracy: 0.5025 - val_loss: 0.6855 - val_accuracy: 0.5546
Epoch 5/10
20/20 [==============================] - 33s 2s/step - loss: 0.6906 - accuracy: 0.5025 - val_loss: 0.6862 - val_accuracy: 0.5038
Epoch 6/10
20/20 [==============================] - 33s 2s/step - loss: 0.6867 - accuracy: 0.5482 - val_loss: 0.6811 - val_accuracy: 0.6688
Epoch 7/10
20/20 [==============================] - 33s 2s/step - loss: 0.6836 - accuracy: 0.5660 - val_loss: 0.6867 - val_accuracy: 0.4849
Epoch 8/10
20/20 [==============================] - 29s 2s/step - loss: 0.6837 - accuracy: 0.5431 - val_loss: 0.6727 - val_accuracy: 0.6506
Epoch 9/10
20/20 [==============================] - 30s 2s/step - loss: 0.6766 - accuracy: 0.5838 - val_loss: 0.6720 - val_accuracy: 0.5684
Epoch 10/10
20/20 [==============================] - 33s 2s/step - loss: 0.6812 - accuracy: 0.5609 - val_loss: 0.6811 - val_accuracy: 0.4981
```

```
# Получаем потери на тестовом наборе данных после аугментации
test_losses = history_additional.history['val_loss']

# Вычисляем среднее значение потерь
mean_test_loss = sum(test_losses) / len(test_losses)

print("Среднее значение потерь на тестовом наборе данных после аугментации:", mean_test_loss)
```

- Среднее значение потерь на тестовом наборе данных после аугментации: 0.68359015583992

**Вопрос 6**

Каково среднее значение точности на тестовом наборе данных за последние 5 эпох (с 6 по 10) после аугментации?

- 0.84
- 0.54 (мой ответ 0,57)
- 0.44
- 0.24

```
# Получаем точность на тестовом наборе данных после аугментации
test_accuracies = history_additional.history['val_accuracy']

# Выбираем точности за последние 5 эпох (индексы с 5 по 9 включительно)
last_5_epoch_test_accuracies = test_accuracies[5:]

# Вычисляем среднее значение точности за последние 5 эпох
mean_test_accuracy_last_5_epochs = sum(last_5_epoch_test_accuracies) / len(last_5_epoch_test_accuracies)

print("Среднее значение точности на тестовом наборе данных за последние 5 эпох после аугментации:", mean_test_accuracy_last_5_epochs)
```

```
Среднее значение точности на тестовом наборе данных за последние 5 эпох после аугментации: 0.5741530776023864
```