Отчет (Этап 4).

1. Тема, описание задачи.

Тема диплома: "Сегментация лица".
Описание задачи: Создать нейросеть, которая будет осуществлять сегментацию частей лица по фотографии.Провести сегментацию для каждого класса на фотографии (глаза, нос, рот, уши, лоб)

2. База данных.

https://drive.google.com/file/d/1OUA5iW0lOOcnlrCYMDImotpGI8sbysDA/view?usp=sharing
Датасет “datasett.zip”. В нем 3 папки: test, train, val. В каждой из этих папок есть 2 папки: images - 2000 фотографий лиц в формате jpg и labels - 2000 масок с сегментацией частей лица в формате png. Всего 11 классов.

3. Параметризация данных.

Выбран размер изображений - 128,128.
Размер батча - 32.
Количество эпох - 33.
Скорость обучения - 0.001.
Путь к моему датасету - ‘/content/datasett’

4. Архитектура нейросети.

Выбрана простая модель Unet со слоями Batch Normalization и Dropout.

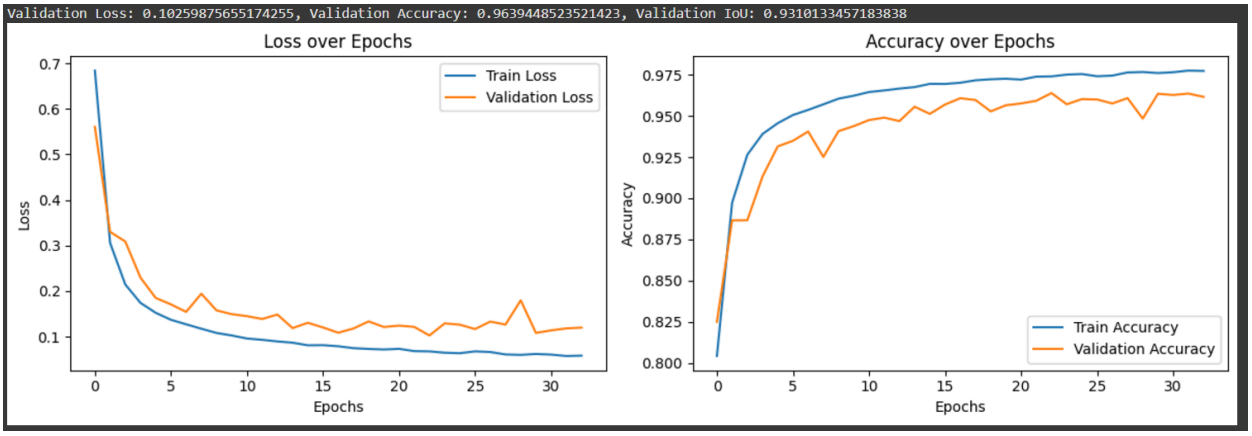
```
def unet_model(input_shape, num_classes, dropout_rate=0.4):
    inputs = layers.Input(shape=input_shape)

    # Encoder (Сжатие)
    c1 = layers.Conv2D(32, (3, 3), padding='same')(inputs)
    c1 = layers.BatchNormalization()(c1)  # Добавлена Batch Normalization
    c1 = layers.Activation('relu')(c1)
    c1 = layers.Conv2D(32, (3, 3), padding='same')(c1)
    c1 = layers.BatchNormalization()(c1)  # Добавлена Batch Normalization
    c1 = layers.Activation('relu')(c1)
    c1 = layers.Dropout(dropout_rate)(c1)
    p1 = layers.MaxPooling2D((2, 2))(c1)

    # Уровень 2
    c2 = layers.Conv2D(64, (3, 3), padding='same')(p1)
    c2 = layers.BatchNormalization()(c2)  # Добавлена Batch Normalization
    c2 = layers.Activation('relu')(c2)
```

```
    c2 = layers.Conv2D(64, (3, 3), padding='same')(c2)
    c2 = layers.BatchNormalization()(c2)  # Добавлена Batch Normalization
    c2 = layers.Activation('relu')(c2)
    c2 = layers.Dropout(dropout_rate)(c2)
    p2 = layers.MaxPooling2D((2, 2))(c2)

    # Уровень 3
    c3 = layers.Conv2D(128, (3, 3), padding='same')(p2)
    c3 = layers.BatchNormalization()(c3)  # Добавлена Batch Normalization
    c3 = layers.Activation('relu')(c3)
    c3 = layers.Conv2D(128, (3, 3), padding='same')(c3)
    c3 = layers.BatchNormalization()(c3)  # Добавлена Batch Normalization
    c3 = layers.Activation('relu')(c3)
    c3 = layers.Dropout(dropout_rate)(c3)
    p3 = layers.MaxPooling2D((2, 2))(c3)

    # Уровень 4
    c4 = layers.Conv2D(256, (3, 3), padding='same')(p3)
    c4 = layers.BatchNormalization()(c4)  # Добавлена Batch Normalization
    c4 = layers.Activation('relu')(c4)
    c4 = layers.Conv2D(256, (3, 3), padding='same')(c4)
    c4 = layers.BatchNormalization()(c4)  # Добавлена Batch Normalization
    c4 = layers.Activation('relu')(c4)
    c4 = layers.Dropout(dropout_rate)(c4)
    p4 = layers.MaxPooling2D((2, 2))(c4)

    # Bottleneck (бутылочное горлышко)
    c5 = layers.Conv2D(512, (3, 3), padding='same')(p4)
    c5 = layers.BatchNormalization()(c5)  # Добавлена Batch Normalization
    c5 = layers.Activation('relu')(c5)
    c5 = layers.Conv2D(512, (3, 3), padding='same')(c5)
    c5 = layers.BatchNormalization()(c5)  # Добавлена Batch Normalization
    c5 = layers.Activation('relu')(c5)
    c5 = layers.Dropout(dropout_rate)(c5)
```

```
    # Декодирование
    u6 = layers.Conv2DTranspose(256, (2, 2), strides=(2, 2), padding='same')(c5)
    u6 = layers.concatenate([u6, c4])
    c6 = layers.Conv2D(256, (3, 3), padding='same')(u6)
    c6 = layers.BatchNormalization()(c6)  # Добавлена Batch Normalization
    c6 = layers.Activation('relu')(c6)
    c6 = layers.Conv2D(256, (3, 3), padding='same')(c6)
    c6 = layers.BatchNormalization()(c6)  # Добавлена Batch Normalization
    c6 = layers.Activation('relu')(c6)
    c6 = layers.Dropout(dropout_rate)(c6)

    u7 = layers.Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same')(c6)
    u7 = layers.concatenate([u7, c3])
    c7 = layers.Conv2D(128, (3, 3), padding='same')(u7)
    c7 = layers.BatchNormalization()(c7)  # Добавлена Batch Normalization
    c7 = layers.Activation('relu')(c7)
    c7 = layers.Conv2D(128, (3, 3), padding='same')(c7)
    c7 = layers.BatchNormalization()(c7)  # Добавлена Batch Normalization
    c7 = layers.Activation('relu')(c7)
    c7 = layers.Dropout(dropout_rate)(c7)

    u8 = layers.Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same')(c7)
    u8 = layers.concatenate([u8, c2])
    c8 = layers.Conv2D(64, (3, 3), padding='same')(u8)
    c8 = layers.BatchNormalization()(c8)  # Добавлена Batch Normalization
    c8 = layers.Activation('relu')(c8)
    c8 = layers.Conv2D(64, (3, 3), padding='same')(c8)
    c8 = layers.BatchNormalization()(c8)  # Добавлена Batch Normalization
    c8 = layers.Activation('relu')(c8)
    c8 = layers.Dropout(dropout_rate)(c8)

    u9 = layers.Conv2DTranspose(32, (2, 2), strides=(2, 2), padding='same')(c8)
    u9 = layers.concatenate([u9, c1])
    c9 = layers.Conv2D(32, (3, 3), padding='same')(u9)
    c9 = layers.BatchNormalization()(c9)  # Добавлена Batch Normalization
```

```
c9 = layers.Activation('relu')(c9)

c9 = layers.Conv2D(32, (3, 3), padding='same')(c9)

c9 = layers.BatchNormalization()(c9)  # Добавлена Batch Normalization

c9 = layers.Activation('relu')(c9)

c9 = layers.Dropout(dropout_rate)(c9)


outputs = layers.Conv2D(num_classes, (1, 1), activation='softmax')(c9)
```

5. Графическое подтверждение.



6. Ноутбук.

1) Диплом этап 5 черновик.
2) Диплом этап 5 итог.

7. Выводы.

Тренировочная точность на первых эпохах начинает с уровня 70.56% и достигает 97.8% на 33-й эпохе. Уровень IoU (Intersection over Union), который является важной метрикой для задач семантической сегментации, показывает аналогичный тренд, начиная с 0.5870 и достигая 0.9574 на 33-й эпохе.

Потеря (loss) снижается практически на всех этапах, что указывает на то, что модель учится эффективно. Потеря на валидации также снижается, что подтверждает, что модель обобщает до некоторой степени.

Вывод: Высокие значения точности и IoU на валидационном наборе (более 96%) показывают, что модель хорошо справляется с задачей сегментации данных. Устойчивое снижение потерь указывает на то, что модель хорошо обучается и адаптируется к данным.

8. План дальнейшей работы.

Попробовать оптимизировать модель: добавить L2-регуляризацию, чтобы уменьшить риск переобучения.