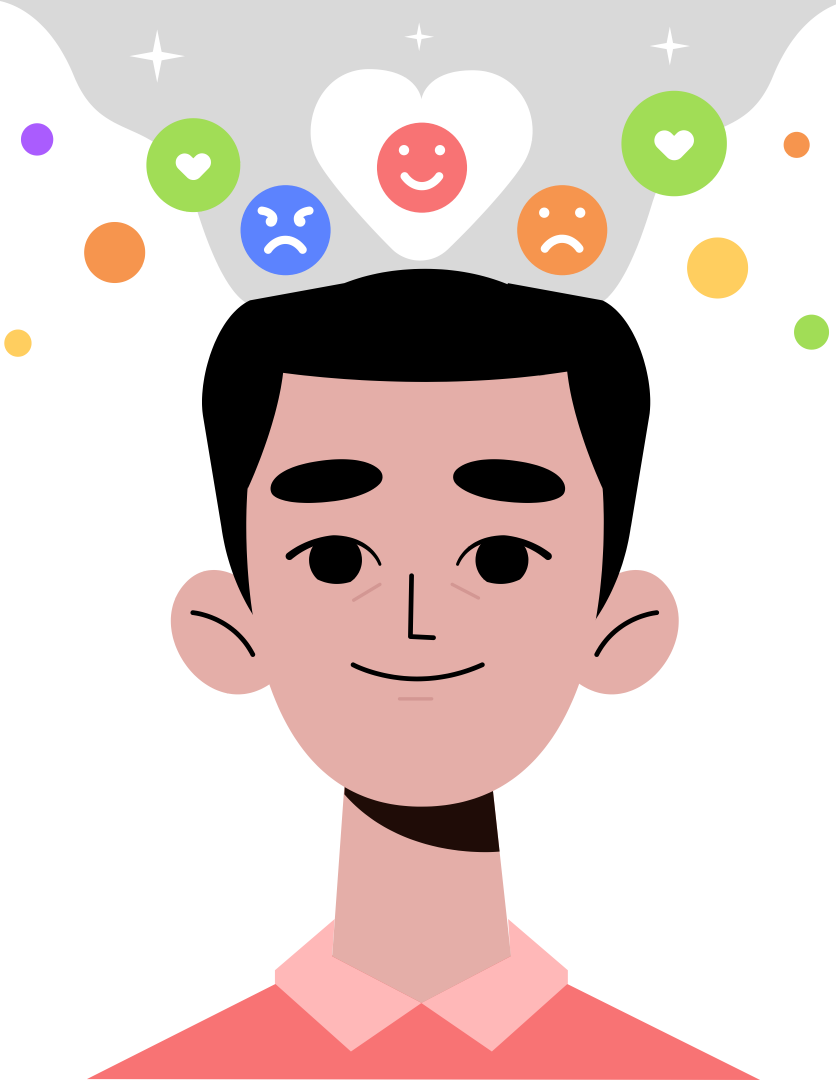


Chatbot psychologist

НЕЙРОТЕХНОЛОГИИ И
АФФЕКТИВНЫЕ ВЫЧИСЛЕНИЯ

Кремпольская Е.А.
Касьяненко В.М.

Датасет для обучения модели для определения эмоций по голосу

CREMA-D: Crowd-sourced Emotional Multimodal Actors Dataset

Датасет содержит более 7 400 аудиозаписей с актерами, произносящими фразы с различными эмоциями (нейтральность, злость, отвращение, страх, счастье и грусть). Эмоции выражаются голосом, мимикой и интонацией. Данные предназначены для анализа эмоций по аудиоканалу, видео и тексту.

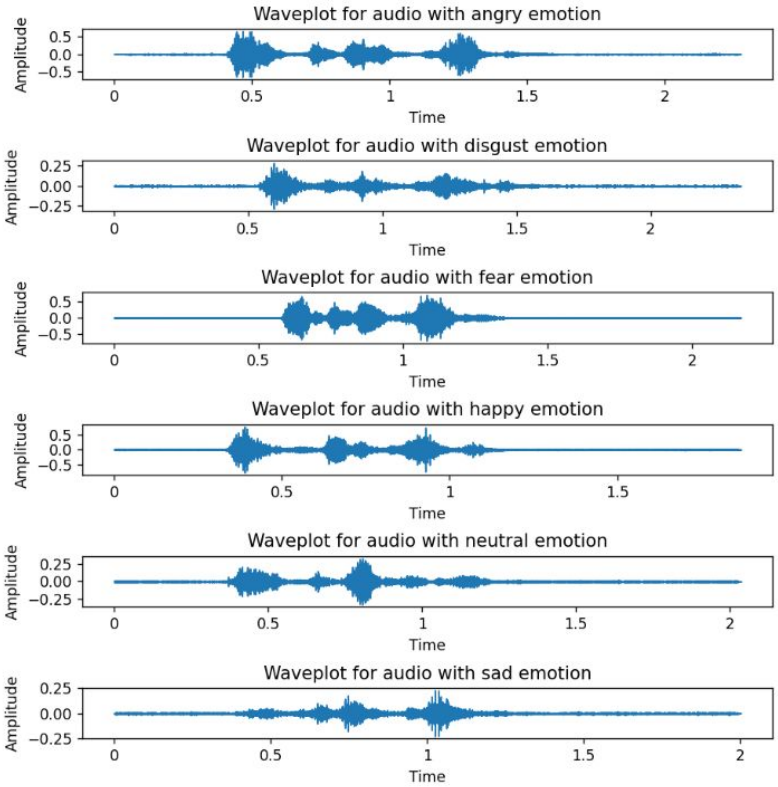
RAVDESS: Ryerson Audio-Visual Database of Emotional Speech and Song

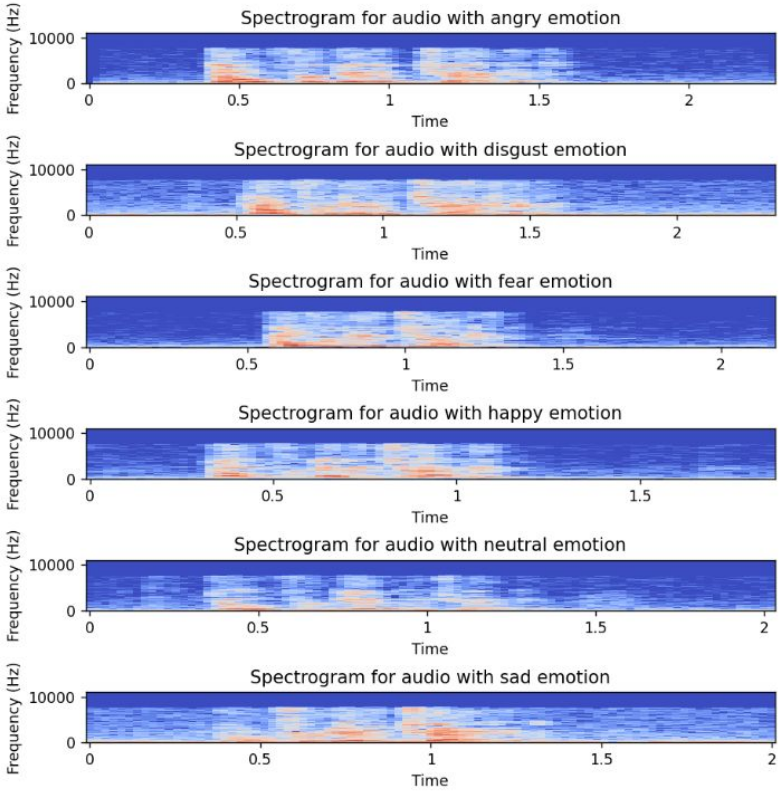
Высококачественный датасет эмоциональной речи и пения от профессиональных актеров. Содержит 1 440 аудиофайлов, в которых выражены такие эмоции, как спокойствие, счастье, грусть, злость, страх, отвращение и удивление. Используется для распознавания эмоций по речи и звуку.

TESS: Toronto Emotional Speech Set

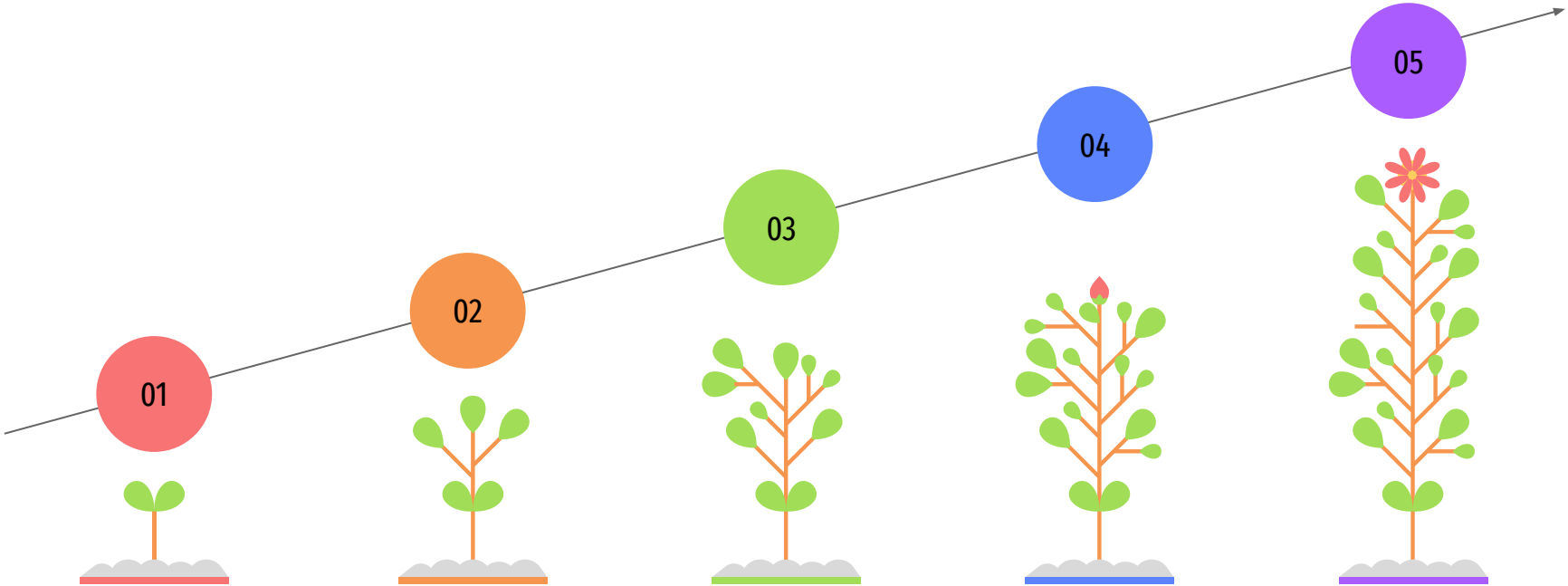
Датасет включает около 2 800 аудиофайлов с речью, произнесённой двумя женщинами в возрасте старше 60 лет. Каждое высказывание озвучено с одной из семи эмоций: гнев, отвращение, страх, счастье, нейтральность, печаль и удивление. Подходит для задач классификации эмоций на основе речи.

Различные эмоции отражаются в амплитуде и частотах аудиосигнала





Подготовка данных для обучения



Чтение данных

Подготовка признаков и меток

Разделение на обучающие и тестовые данные

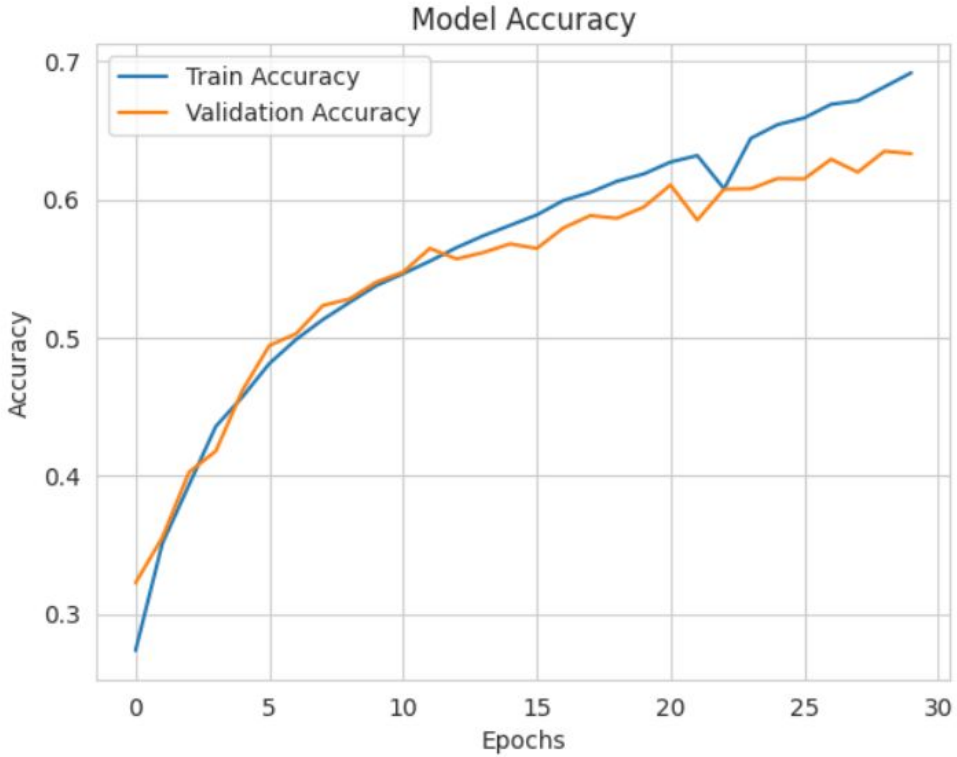
Масштабирование признаков

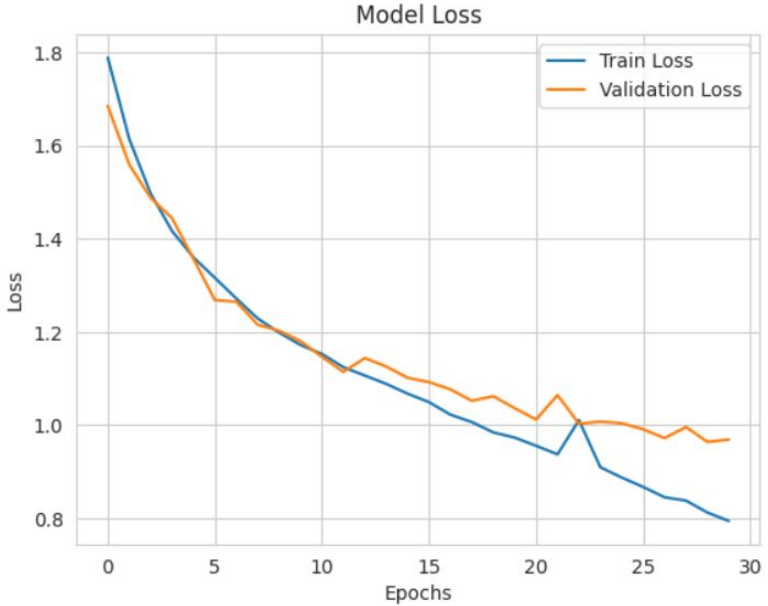
Результат

Модель

```
model = tf.keras.Sequential([
    Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=(x_train.shape[1], x_tra
in.shape[2])),
    MaxPooling1D(pool_size=2),
    Dropout(0.2),
    Conv1D(filters=128, kernel_size=3, activation='relu'),
    MaxPooling1D(pool_size=2),
    Dropout(0.3),
    Bidirectional(LSTM(128, return_sequences=True)),
    Bidirectional(LSTM(64)),
    Dense(128, activation='relu'),
    Dropout(0.4),
    Dense(8, activation='softmax')
])

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

Результаты


Model Accuracy
Train Accuracy
Validation Accuracy
Accuracy
Epochs
0.3
0.4
0.5
0.6
0.7
0
5
10
15
20
25
30


Model Loss
Train Loss
Validation Loss
Loss
Epochs
0.8
1.0
1.2
1.4
1.6
1.8
0
5
10
15
20
25
30

Датасет для обучения модели для определения эмоций в тексте

Emotions Dataset for NLP

Текстовый датасет, созданный для задач анализа эмоций в естественном языке. Содержит около 20 000 коротких предложений на английском языке, размеченных по одной из шести категорий эмоций: радость, печаль, гнев, страх, любовь и удивление. Используется для обучения и тестирования моделей распознавания эмоций по тексту. Подходит для классификации эмоций и других NLP-задач.

Подготовка данных для обучения

- приведение к нижнему регистру
- удаление пунктуации и спец-символов
- удаление стоп-слов
- лемматизация
- разделение данных

Dense NN

```
embedding_1 = tf.keras.layers.Embedding(
    input_dim=len(text_vectorizer.get_vocabulary()),
    output_dim=128,
    embeddings_initializer='uniform',
    input_length=15,
    name='embedding'
)

inputs = tf.keras.layers.Input(shape=(1,), dtype='string')

x = text_vectorizer(inputs)
x = embedding_1(x)
x = tf.keras.layers.GlobalAveragePooling1D()(x)

outputs = tf.keras.layers.Dense(6, activation='softmax')(x)

model_1 = tf.keras.Model(inputs, outputs, name='model_1_simple_dense')

model_1.compile(
    optimizer='Adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

model_1_history = model_1.fit(
    train_sentences,
    train_labels,
    epochs=5,
    validation_data=(val_sentences, val_labels),
)
```

```
Epoch 1/5
500/500 [==============================] - 9s 16ms/step - loss: 1.5061 - accuracy: 0.4352 - va
l_loss: 1.3215 - val_accuracy: 0.5365
Epoch 2/5
500/500 [==============================] - 8s 16ms/step - loss: 1.0574 - accuracy: 0.6664 - va
l_loss: 0.9133 - val_accuracy: 0.7375
Epoch 3/5
500/500 [==============================] - 8s 16ms/step - loss: 0.6461 - accuracy: 0.8173 - va
l_loss: 0.7096 - val_accuracy: 0.7790
Epoch 4/5
500/500 [==============================] - 8s 15ms/step - loss: 0.4452 - accuracy: 0.8713 - va
l_loss: 0.6574 - val_accuracy: 0.7795
Epoch 5/5
500/500 [==============================] - 8s 15ms/step - loss: 0.3455 - accuracy: 0.8982 - va
l_loss: 0.6563 - val_accuracy: 0.7780
```

```
model_1.evaluate(test_sentences, test_labels)
```

```
63/63 [==============================] - 0s 2ms/step - loss: 0.6563 - accuracy: 0.7780
```

LSTM

Model 2: LSTM

```
]:
embedding_2 = tf.keras.layers.Embedding(
    input_dim=len(text_vectorizer.get_vocabulary()),
    output_dim=128,
    embeddings_initializer='uniform',
    input_length=15,
    name='embedding_2'
)

inputs = tf.keras.layers.Input(shape=(1,), dtype='string')

x = text_vectorizer(inputs)
x = embedding_2(x)
x = tf.keras.layers.LSTM(64)(x)

outputs = tf.keras.layers.Dense(6, activation='softmax')(x)

model_2 = tf.keras.Model(inputs, outputs, name='model_2_simple_lstm')

model_2.compile(
    optimizer='Adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

model_2_history = model_2.fit(
    train_sentences,
    train_labels,
    epochs=5,
    validation_data=(val_sentences, val_labels)
)
```

```
Epoch 1/5
500/500 [==============================] - 15s 25ms/step - loss: 1.1250 - accuracy: 0.5721 - v
al_loss: 0.6939 - val_accuracy: 0.7540
Epoch 2/5
500/500 [==============================] - 12s 25ms/step - loss: 0.5142 - accuracy: 0.8263 - v
al_loss: 0.6253 - val_accuracy: 0.7745
Epoch 3/5
500/500 [==============================] - 12s 24ms/step - loss: 0.3399 - accuracy: 0.8839 - v
al_loss: 0.6854 - val_accuracy: 0.7800
Epoch 4/5
500/500 [==============================] - 13s 26ms/step - loss: 0.2488 - accuracy: 0.9137 - v
al_loss: 0.7478 - val_accuracy: 0.7665
Epoch 5/5
500/500 [==============================] - 12s 25ms/step - loss: 0.1851 - accuracy: 0.9326 - v
al_loss: 0.8540 - val_accuracy: 0.7655
```

```
model_2.evaluate(test_sentences, test_labels)
```

```
63/63 [==============================] - 0s 5ms/step - loss: 0.8540 - accuracy: 0.7655
```

GRU

Model 3: GRU

```
embedding_3 = tf.keras.layers.Embedding(
    input_dim=len(text_vectorizer.get_vocabulary()),
    output_dim=128,
    embeddings_initializer='uniform',
    input_length=15,
    name='embedding_3'
)

inputs = tf.keras.layers.Input(shape=(1,), dtype='string')

x = text_vectorizer(inputs)
x = embedding_3(x)
x = tf.keras.layers.GRU(64, kernel_regularizer=tf.keras.regularizers.l2(0.01))(x)

outputs = tf.keras.layers.Dense(6, activation='softmax')(x)

model_3 = tf.keras.Model(inputs, outputs, name='model_2_simple_gru')

model_3.compile(
    optimizer='Adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

history_3 = model_3.fit(
    train_sentences,
    train_labels,
    epochs=5,
    validation_data=(val_sentences, val_labels)
)
```

```
Epoch 1/5
500/500 [==============================] - 15s 25ms/step - loss: 1.5779 - accuracy: 0.3808 - v
al_loss: 1.2333 - val_accuracy: 0.5170
Epoch 2/5
500/500 [==============================] - 12s 23ms/step - loss: 0.8344 - accuracy: 0.7053 - v
al_loss: 0.7135 - val_accuracy: 0.7435
Epoch 3/5
500/500 [==============================] - 12s 23ms/step - loss: 0.5326 - accuracy: 0.8265 - v
al_loss: 0.6709 - val_accuracy: 0.7715
Epoch 4/5
500/500 [==============================] - 12s 24ms/step - loss: 0.4285 - accuracy: 0.8634 - v
al_loss: 0.6970 - val_accuracy: 0.7630
Epoch 5/5
500/500 [==============================] - 12s 23ms/step - loss: 0.3638 - accuracy: 0.8852 - v
al_loss: 0.7284 - val_accuracy: 0.7545
```

```
model_3.evaluate(test_sentences, test_labels)
```

```
63/63 [==============================] - 0s 5ms/step - loss: 0.7284 - accuracy: 0.7545
```

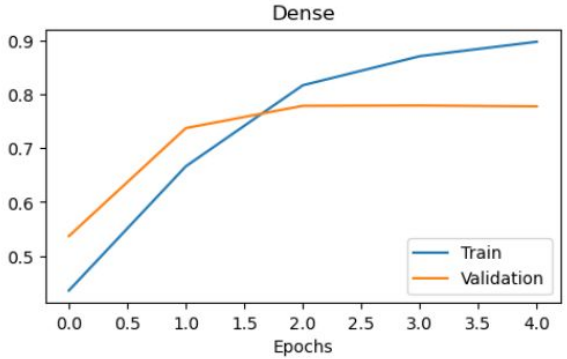
CNN

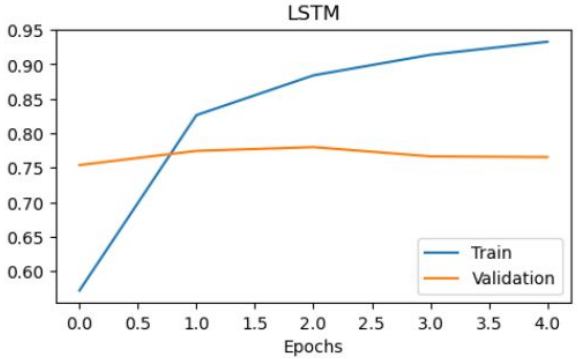
```
embedding_4 = tf.keras.layers.Embedding(
    input_dim=len(text_vectorizer.get_vocabulary()),
    output_dim=128,
    embeddings_initializer='uniform',
    input_length=15,
    name='embedding_4'
)

inputs = tf.keras.layers.Input(shape=(1,), dtype='string')

x = text_vectorizer(inputs)
x = embedding_4(x)
x = tf.keras.layers.Conv1D(
    filters=32,
    kernel_size=5,
    activation='relu',
    kernel_regularizer=tf.keras.regularizers.l2(0.01)
)(x)
x = tf.keras.layers.GlobalMaxPooling1D()(x)

outputs = tf.keras.layers.Dense(6, activation='softmax')(x)

model_4 = tf.keras.Model(inputs, outputs, name='model_4_simple_cnn')

model_4.compile(
    optimizer=tf.keras.optimizers.Adam(),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

model_4_history = model_4.fit(
    train_sentences,
    train_labels,
    epochs=5,
    validation_data=(val_sentences, val_labels)
)
```

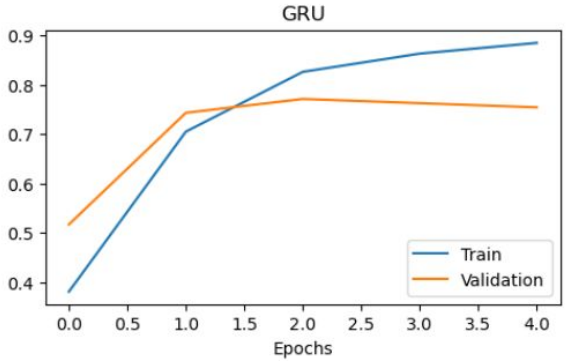
```
Epoch 1/5
500/500 [==============================] - 10s 17ms/step - loss: 1.5124 - accuracy: 0.4498 - v
al_loss: 1.0418 - val_accuracy: 0.7145
Epoch 2/5
500/500 [==============================] - 9s 17ms/step - loss: 0.8380 - accuracy: 0.7496 - va
l_loss: 0.7674 - val_accuracy: 0.7700
Epoch 3/5
500/500 [==============================] - 8s 17ms/step - loss: 0.6232 - accuracy: 0.8232 - va
l_loss: 0.7033 - val_accuracy: 0.7730
Epoch 4/5
500/500 [==============================] - 8s 16ms/step - loss: 0.5092 - accuracy: 0.8639 - va
l_loss: 0.6916 - val_accuracy: 0.7740
Epoch 5/5
500/500 [==============================] - 8s 17ms/step - loss: 0.4352 - accuracy: 0.8874 - va
l_loss: 0.6863 - val_accuracy: 0.7765
```

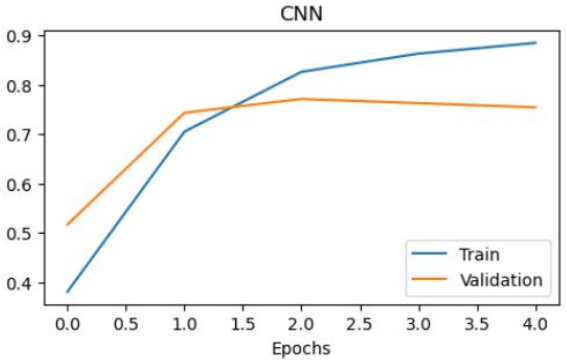
```
model_4.evaluate(test_sentences, test_labels)
```

Результаты









Models

Model 1: Dense NN

Model 2: LSTM

Model 3: GRU

Model 4: CNN