Министерство цифрового развития

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Кафедра прикладной математики и кибернетики

Отчёт

по лабораторной работе № 6 «Нейронные сети с использованием Pytorch для классификации изображений»

Выполнил:

студенты группы ИП-216

Русецкий А.С.

Ливтинов А.Е.
ФИО студента

Работу проверил: Преподаватель
должность преподавателя
Сороковых Д.А.
ФИО преподавателя

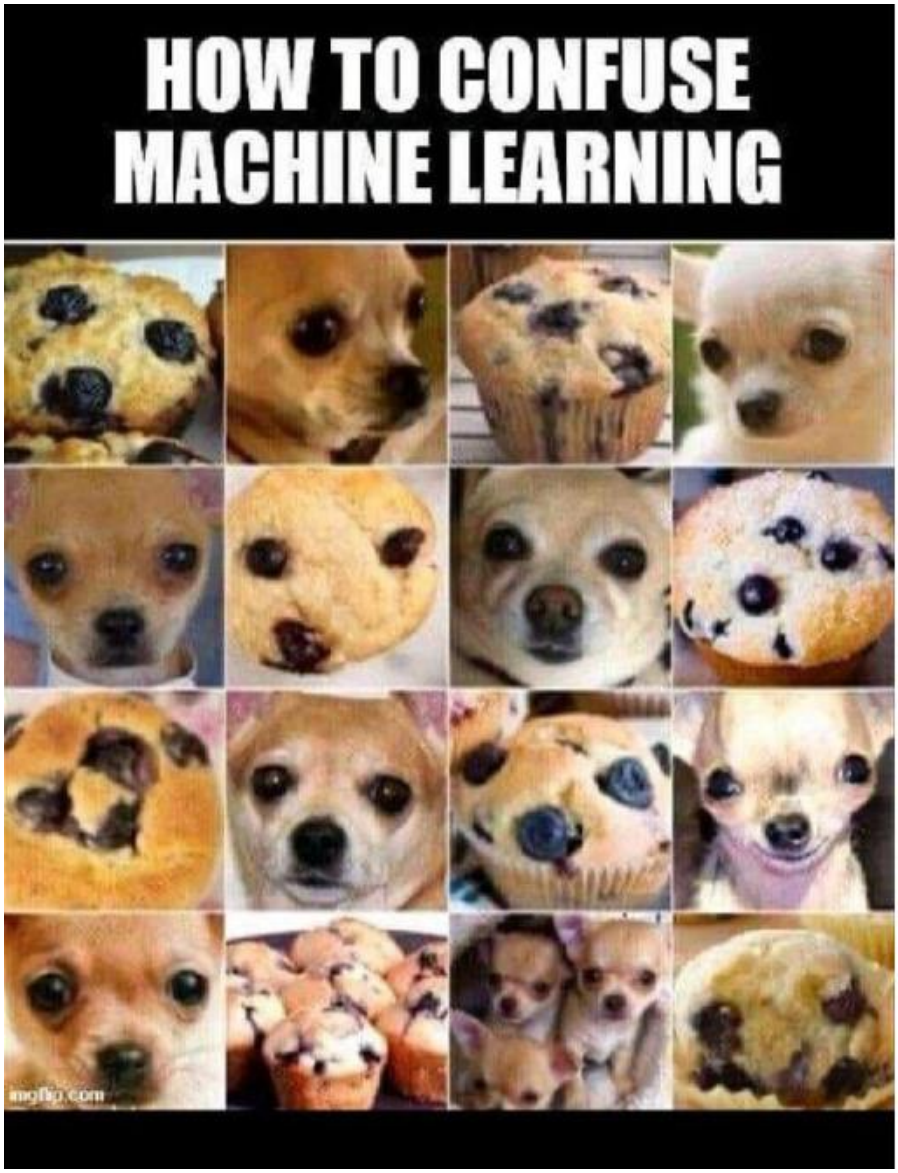
Новосибирск 2025 г.

Задание

Набор данных

Dataset Muffin vs chihuahua

Пример изображений в датасете



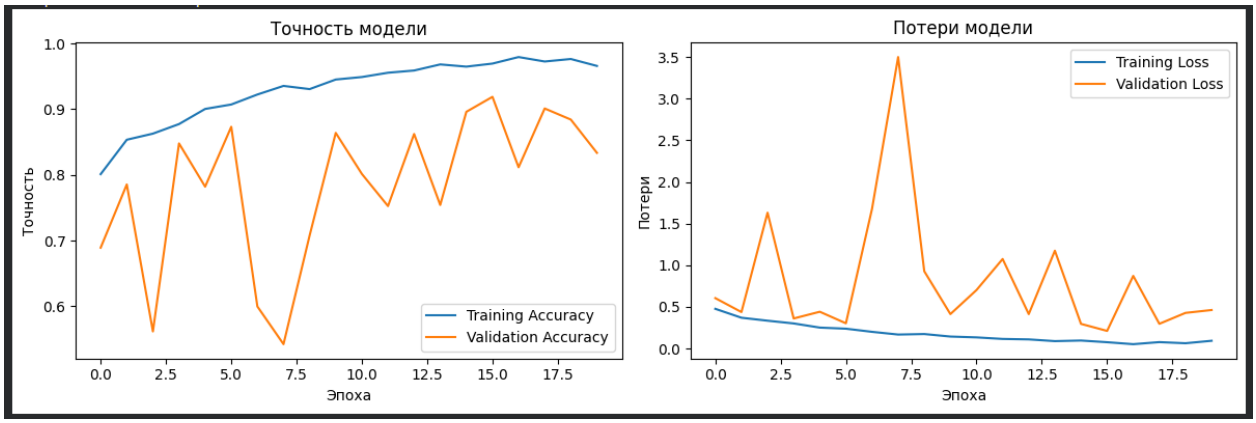
Архитектура CNN

```python
print("\n=== ПРЕДОБРАБОТКА ДАННЫХ ===")
X_train_processed = X_train.astype('float32') / 255.0
X_test_processed = X_test.astype('float32') / 255.0

y_train_processed = keras.utils.to_categorical(y_train, 2)
y_test_processed = keras.utils.to_categorical(y_test, 2)

print(f"X_train shape: {X_train_processed.shape}")
print(f"y_train shape: {y_train_processed.shape}")


def create_cnn_model(input_shape, num_classes):
    """Создание CNN с 3 сверточными слоями + 2 полносвязных + BatchNormalization"""
    model = keras.Sequential([
        layers.Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=input_shape),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2)),

        layers.Conv2D(64, (3, 3), activation='relu', padding='same'),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2)),

        layers.Conv2D(128, (3, 3), activation='relu', padding='same'),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2)),

        layers.Flatten(),
        layers.Dense(256, activation='relu'),
        layers.BatchNormalization(),
        layers.Dropout(0.5),

        layers.Dense(128, activation='relu'),
        layers.BatchNormalization(),
        layers.Dropout(0.3),

        layers.Dense(num_classes, activation='softmax')
    ])

    return model
```

Обучение и ранняя остановка

```
=== ОБУЧЕНИЕ МОДЕЛИ ===
Epoch 1/20
148/148 ━━━━━━━━━━━━━━━━━━━━ 289s 2s/step - accuracy: 0.7587 - loss: 0.5814 - val_accuracy: 0.6892 - val_loss: 0.6056
Epoch 2/20
148/148 ━━━━━━━━━━━━━━━━━━━━ 286s 2s/step - accuracy: 0.8598 - loss: 0.3585 - val_accuracy: 0.7855 - val_loss: 0.4383
Epoch 3/20
148/148 ━━━━━━━━━━━━━━━━━━━━ 319s 2s/step - accuracy: 0.8579 - loss: 0.3451 - val_accuracy: 0.5617 - val_loss: 1.6315
Epoch 4/20
148/148 ━━━━━━━━━━━━━━━━━━━━ 328s 2s/step - accuracy: 0.8683 - loss: 0.3247 - val_accuracy: 0.8480 - val_loss: 0.3615
Epoch 5/20
148/148 ━━━━━━━━━━━━━━━━━━━━ 316s 2s/step - accuracy: 0.8951 - loss: 0.2659 - val_accuracy: 0.7821 - val_loss: 0.4422
Epoch 6/20
148/148 ━━━━━━━━━━━━━━━━━━━━ 320s 2s/step - accuracy: 0.9084 - loss: 0.2345 - val_accuracy: 0.8733 - val_loss: 0.3036
Epoch 7/20
148/148 ━━━━━━━━━━━━━━━━━━━━ 321s 2s/step - accuracy: 0.9205 - loss: 0.1945 - val_accuracy: 0.5997 - val_loss: 1.6779
Epoch 8/20
148/148 ━━━━━━━━━━━━━━━━━━━━ 325s 2s/step - accuracy: 0.9307 - loss: 0.1682 - val_accuracy: 0.5422 - val_loss: 3.4998
Epoch 9/20
148/148 ━━━━━━━━━━━━━━━━━━━━ 320s 2s/step - accuracy: 0.9371 - loss: 0.1600 - val_accuracy: 0.7078 - val_loss: 0.9284
Epoch 10/20
148/148 ━━━━━━━━━━━━━━━━━━━━ 329s 2s/step - accuracy: 0.9505 - loss: 0.1381 - val_accuracy: 0.8640 - val_loss: 0.4139
Epoch 11/20
148/148 ━━━━━━━━━━━━━━━━━━━━ 284s 2s/step - accuracy: 0.9504 - loss: 0.1251 - val_accuracy: 0.8015 - val_loss: 0.7029
Epoch 12/20
148/148 ━━━━━━━━━━━━━━━━━━━━ 320s 2s/step - accuracy: 0.9564 - loss: 0.1114 - val_accuracy: 0.7525 - val_loss: 1.0756
Epoch 13/20
148/148 ━━━━━━━━━━━━━━━━━━━━ 324s 2s/step - accuracy: 0.9504 - loss: 0.1324 - val_accuracy: 0.8623 - val_loss: 0.4138
Epoch 14/20
148/148 ━━━━━━━━━━━━━━━━━━━━ 291s 2s/step - accuracy: 0.9711 - loss: 0.0878 - val_accuracy: 0.7542 - val_loss: 1.1752
Epoch 15/20
148/148 ━━━━━━━━━━━━━━━━━━━━ 321s 2s/step - accuracy: 0.9680 - loss: 0.0884 - val_accuracy: 0.8961 - val_loss: 0.2966
Epoch 16/20
148/148 ━━━━━━━━━━━━━━━━━━━━ 285s 2s/step - accuracy: 0.9713 - loss: 0.0755 - val_accuracy: 0.9189 - val_loss: 0.2121
Epoch 17/20
148/148 ━━━━━━━━━━━━━━━━━━━━ 326s 2s/step - accuracy: 0.9766 - loss: 0.0591 - val_accuracy: 0.8117 - val_loss: 0.8726
Epoch 18/20
148/148 ━━━━━━━━━━━━━━━━━━━━ 320s 2s/step - accuracy: 0.9814 - loss: 0.0568 - val_accuracy: 0.9012 - val_loss: 0.2974
Epoch 19/20
148/148 ━━━━━━━━━━━━━━━━━━━━ 295s 2s/step - accuracy: 0.9721 - loss: 0.0701 - val_accuracy: 0.8843 - val_loss: 0.4285
Epoch 20/20
148/148 ━━━━━━━━━━━━━━━━━━━━ 304s 2s/step - accuracy: 0.9777 - loss: 0.0573 - val_accuracy: 0.8336 - val_loss: 0.4630
```

Графики



Пример предсказаний

Заключение

Общие результаты

В ходе выполнения лабораторной работы была успешно разработана и обучена сверточная нейронная сеть (CNN) для классификации изображений кексов и чихуахуа. Модель достигла точности 83,36% на тестовой выборке, что является хорошим результатом для данной задачи бинарной классификации.

Архитектура модели

Разработанная CNN содержит 3 сверточных блока с батч-нормализацией и max-pooling слоями, что позволило эффективно извлекать иерархические признаки из изображений. Общее количество обучаемых параметров составило 8,516,482, что обеспечило достаточную выразительность модели для решения поставленной задачи.