Разработка мобильного приложения под iOS с применением технологий машинного обучения

Студент:
Леонова Алина Дмитриевна
НФИбд-02-17

Научный руководитель:
к.ф.-м.н, доцент кафедры ИТ
Шорохов Сергей Геннадьевич

Кафедра информационных технологий

Российский университет дружбы народов

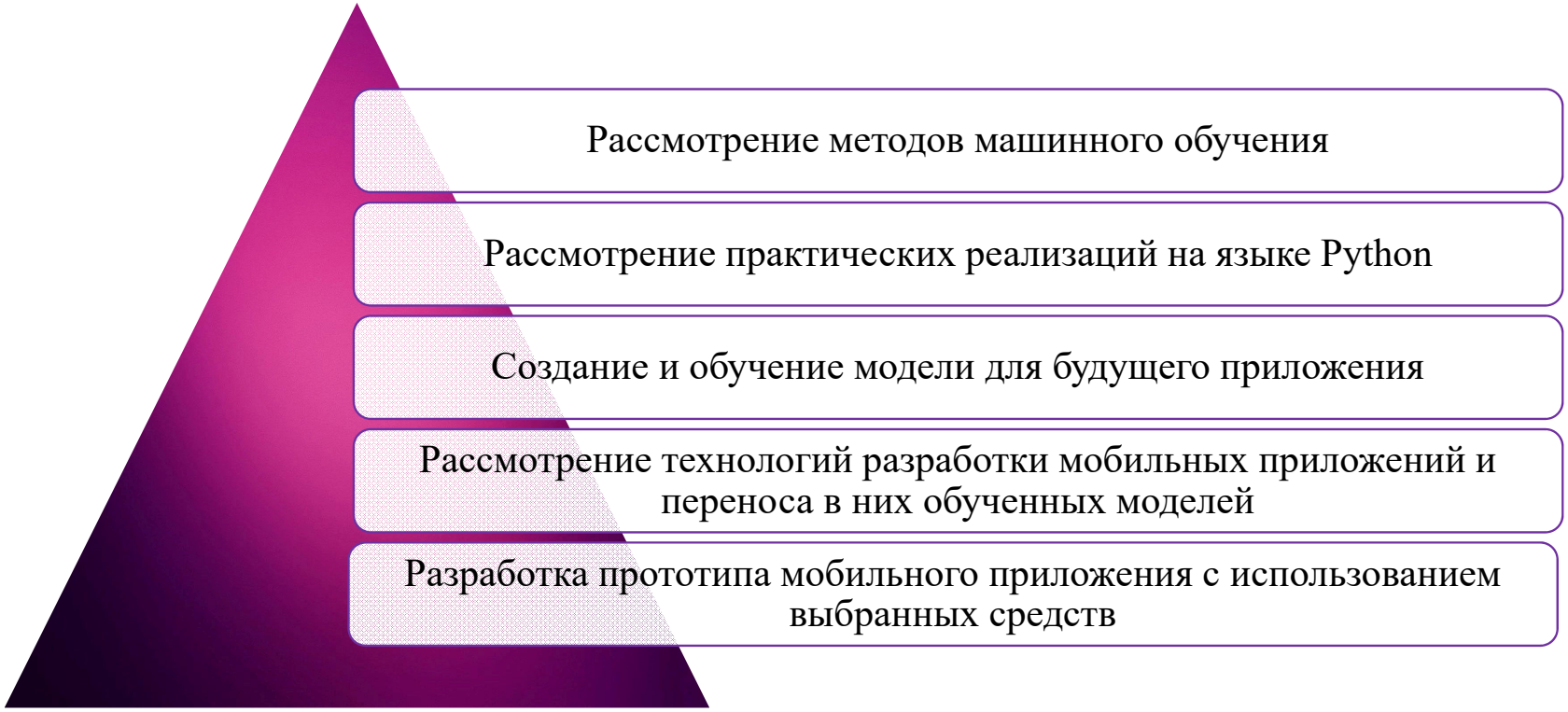
Введение

Машинное обучение — это раздел искусственного интеллекта, изучающий методы построения алгоритмов, способных обучаться.
Его цель — предсказать результат по входным данным.

Мобильные приложения пользуются спросом благодаря своему удобству в освоении и использовании.

Постановка задачи

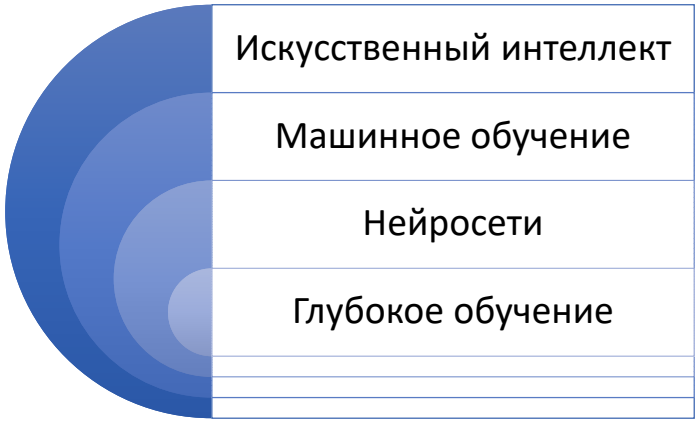
Цель работы: разработка мобильного приложения для iOS, использующего технологии машинного обучения.

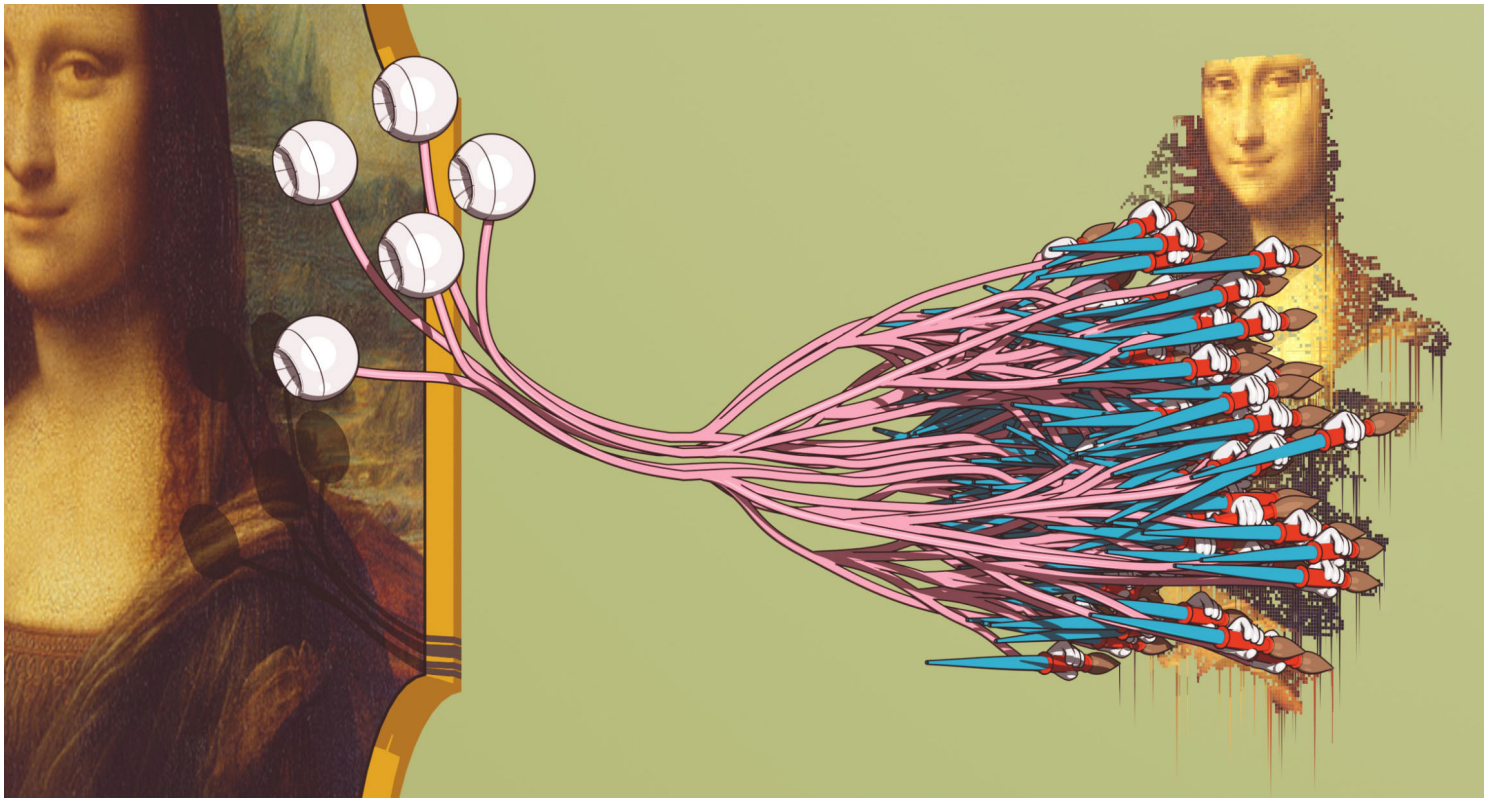


Классификация задач машинного обучения

Глубокое обучение и нейронные сети

Глубокое обучение — это метод машинного обучения с помощью искусственной нейронной сети, имитирующей человеческий мозг, состоящий из нейронов, организованных в сеть.



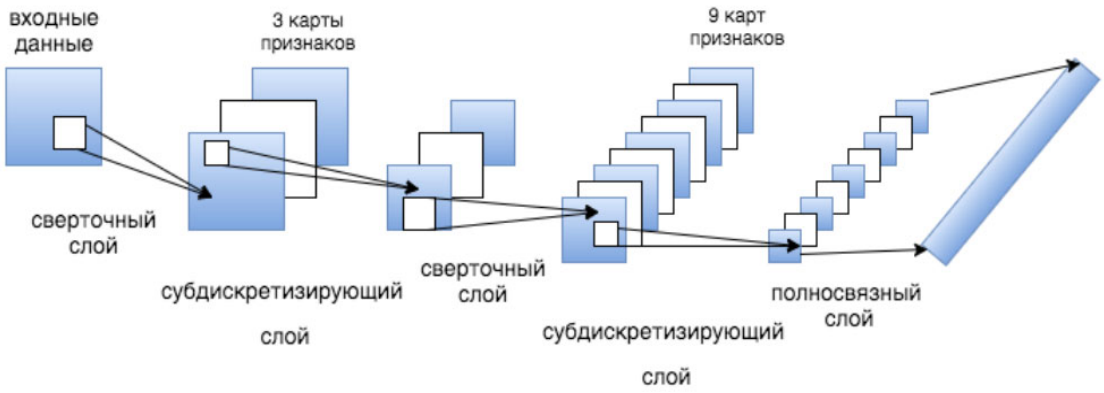


Свёрточные нейронные сети (CNN)

Главная идея: применение одной операции к разным частям изображения.

Нейроны, как в зрительной коре, сгруппированы в слои разных типов.

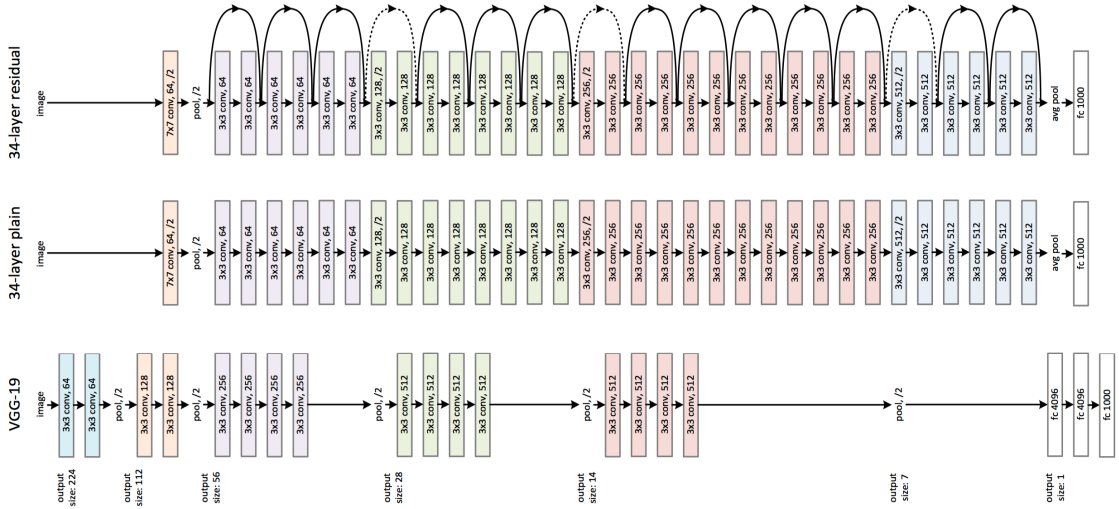
Пример архитектуры свёрточной сети



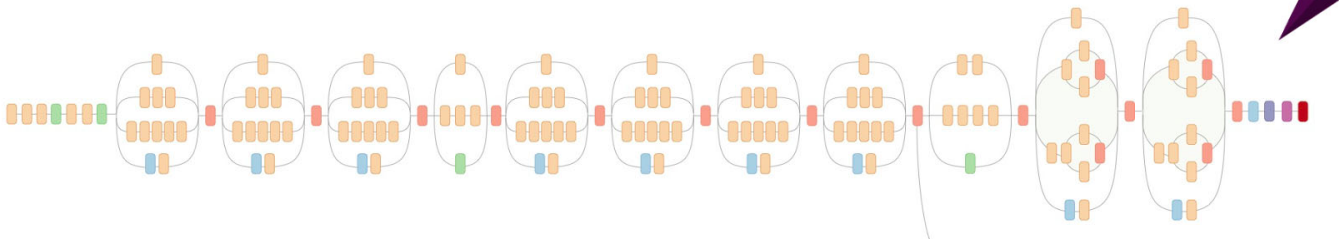
- **свёрточные слои**: активация нейронов следующего уровня — линейная комбинация активаций предыдущего уровня;
- **слои активации**: нелинейная функция активации скалярного результата каждой свёртки, логически объединяются со свёрточными слоями;
- **субдискретизирующие слои**: воспроизводят активацию нейронов предыдущего уровня, уменьшая размеры изображения за счёт процедуры пулинга.

Архитектура сетей (ResNet50, InceptionV3, InceptionResNetV2)

ResNet
2015



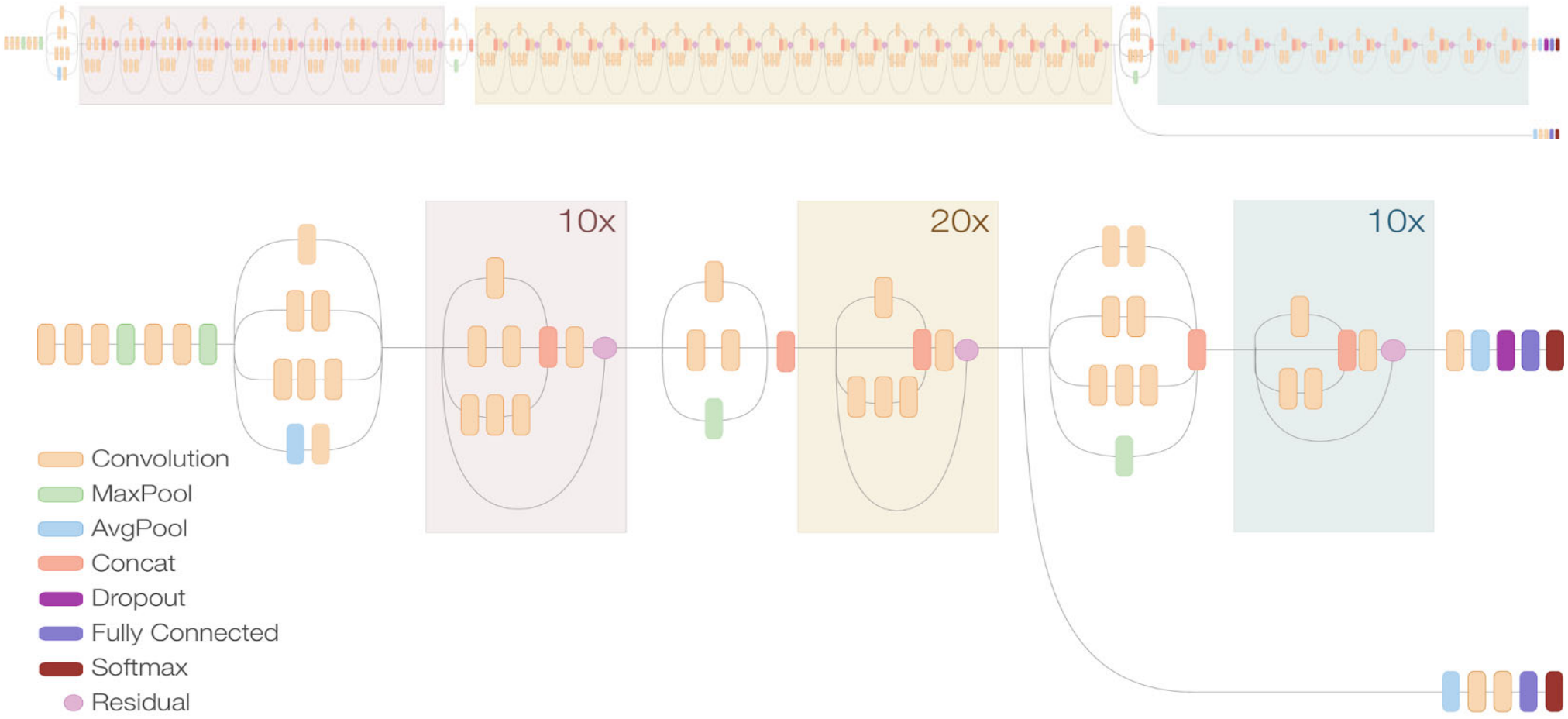
InceptionV3
2016



InceptionResNetV2
2016

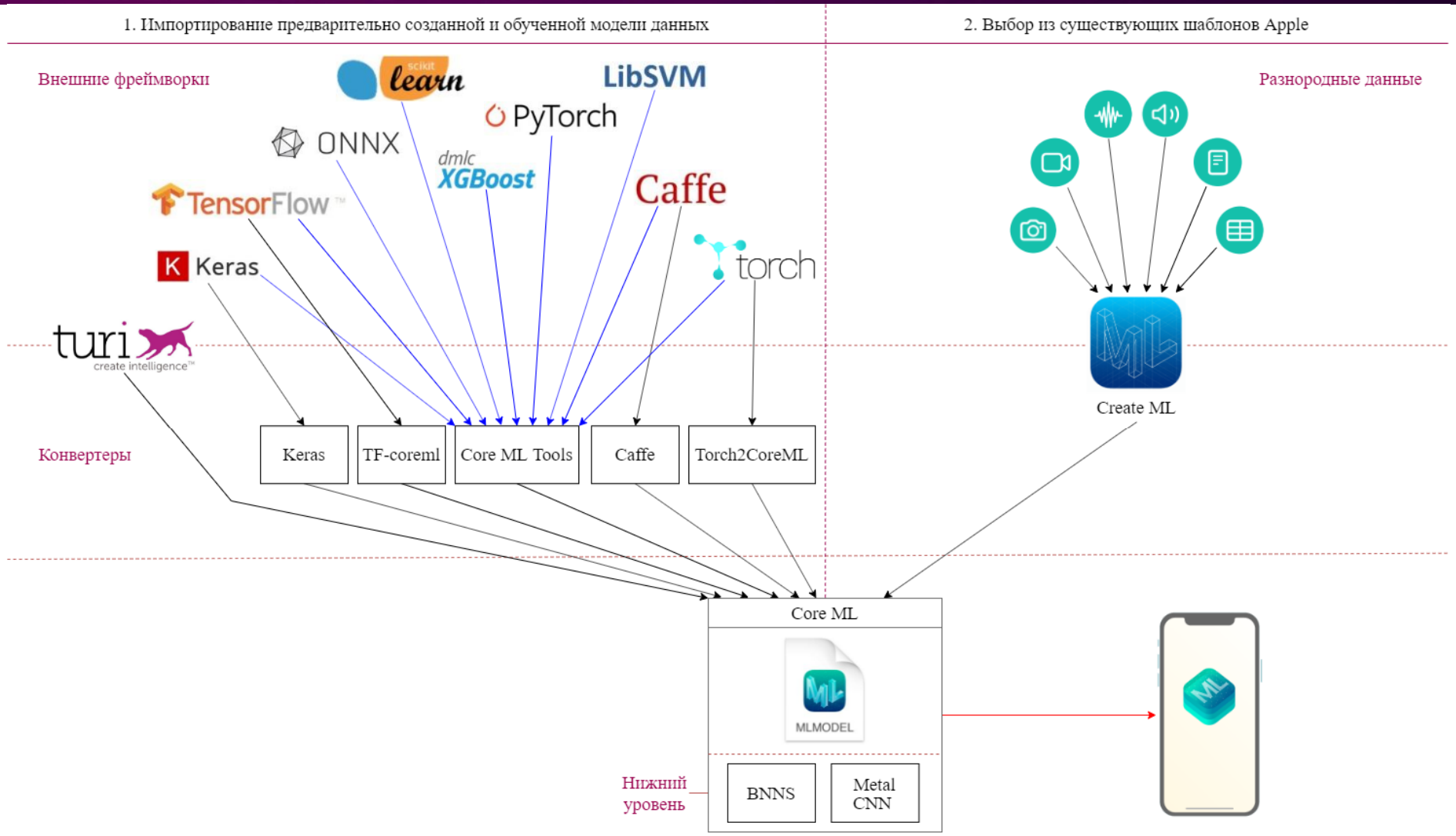
Convolution
AvgPool
MaxPool
Concat
Dropout
Fully connected
Softmax

Архитектура сети InceptionResNetV2



Основные этапы разработки мобильного приложения

Способы реализации модели машинного обучения для iOS-приложений

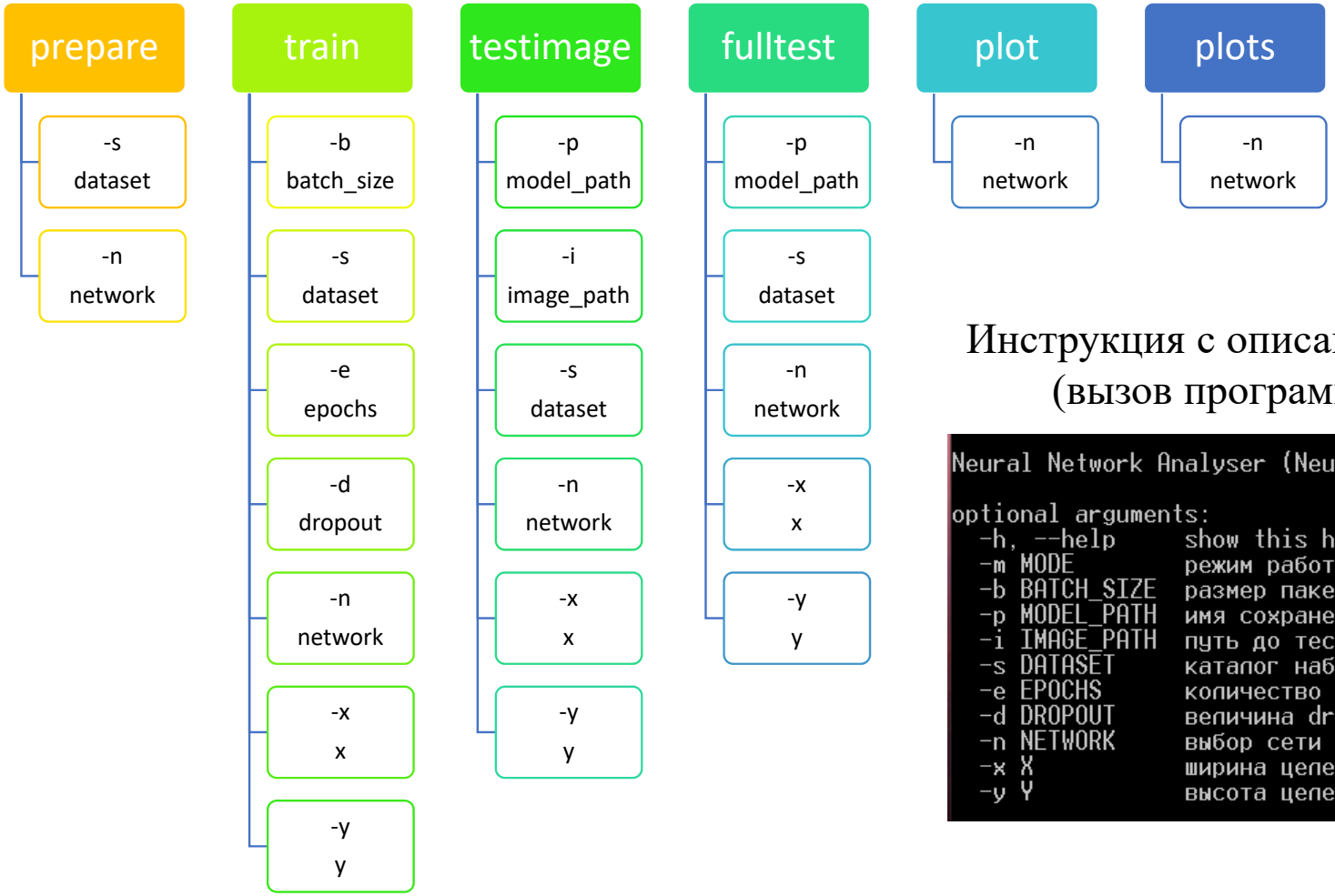


Сравнительный анализ фреймворков

Фреймворк	Основное предназначение	Написан на языках	Поддерживаемые языки	Специальный конвертер в CoreML	Скорость работы	Возможность ускорения	Поддержка нейронных сетей
CreateML	упрощает разработку моделей	Swift	Swift	CreateML	++	GPU (BNNS, Metal CNN)	+
Turi Create	упрощает разработку моделей	Python, C++	Python, C++	Turi Create	+	GPU	-
Keras	работа поверх других фреймворков МО	Python, C++	Python, C++	Keras	+	GPU	+
TensorFlow	классификация образов	Python, C++	Python, C++	TF-coreml	+	GPU, TPU	+
ONNX	глубокое обучение и ускорение логического вывода	Python, C++	Python, C#, C++, Java	—	+	GPU	+
Scikit-learn	для взаимодействия с числовыми и научными библиотеками NumPy и SciPy	Python, Cython, C, C++	Python, C++	—	-	—	+
XGBoost	повышение качества дерева	Python, C++	Python, C++	—	+	GPU	-
PyTorch	глубокое обучение	Python, C++	Python, C++	—	++	GPU	+
LibSVM	методы опорных векторов	Java, C++	Python, R, MATLAB, Perl, Ruby и др.	—	-	—	-
Caffe	классификация и сегментация изображений	C++	Python, C++, MATLAB	Caffe	+	GPU	+
Torch	для глубинного обучения и научных расчётов	C, C++, Lua	C++, Lua	Torch2CoreML	+	GPU	+

Программа для анализа эффективности архитектур нейронных сетей (neurotest.py)

Режимы работы и значимые для них параметры



Инструкция с описанием предусмотренных параметров (вызов программы с параметром –h или –help)

```
Neural Network Analyser (NeuroTestAL)

optional arguments:
  -h, --help      show this help message and exit
  -m MODE         режим работы: prepare/train/testimage/fulltest/plot/plots
  -b BATCH_SIZE   размер пакета
  -p MODEL_PATH   имя сохраненной модели
  -i IMAGE_PATH   путь до тестового изображения
  -s DATASET      каталог набора данных
  -e EPOCHS       количество эпох
  -d DROPOUT      величина dropout
  -n NETWORK      выбор сети (ResNet50, InceptionV3, InceptionResNetV2)
  -x X            ширина целевого изображения
  -y Y            высота целевого изображения
```

Пример работы с Keras и TensorFlow, функция для режима обучения (neurotest.py)

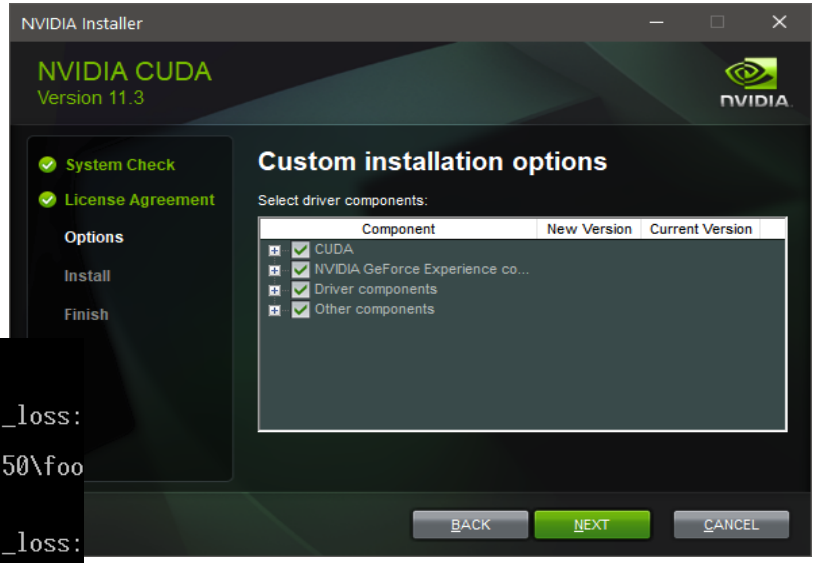
```
def train(args):
    X_train, X_test = setup_generator(make_dataset_path(args, 'train'),
                                      make_dataset_path(args, 'test'), args.batch_size, shape[:2])
    print(X_train)
    callbacks = []
    callbacks.append(ModelCheckpoint(filepath=make_model_path(args, args.dataset +
                                '-weights.epoch-{epoch:02d}-val_loss-{val_loss:.4f}-val_accuracy-
                                    {val_accuracy:.4f}.hdf5'), verbose=1, save_best_only=True))
    callbacks.append(CSVLogger(make_model_path(args, 'history_log.csv')))

    model_final = create_model(args.network, X_train.num_classes, args.dropout, shape)
    train_model(model_final, X_train, X_test, callbacks, args)
```

Пример работы с Keras и TensorFlow, функция для обучения модели (neurotest.py)

```
def train_model(model_final, train_generator, validation_generator, callbacks, args):
    model_final.compile(
        loss='categorical_crossentropy',
        optimizer='adam',
        metrics=['accuracy'])

    model_final.fit_generator(train_generator,
        validation_data=validation_generator,
        epochs=args.epochs, callbacks=callbacks,
        steps_per_epoch=train_generator.samples//args.batch_size,
        validation_steps=validation_generator.samples//args.batch_size)
```

Вывод консоли в процессе обучения

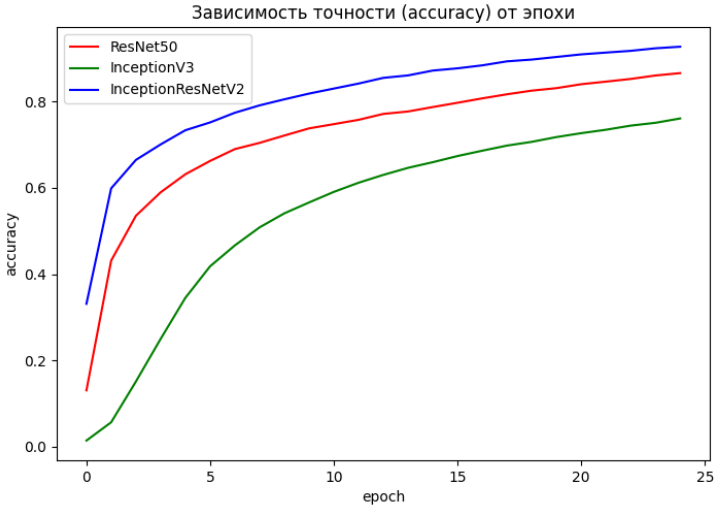
Окно выбора подключения компонент драйвера

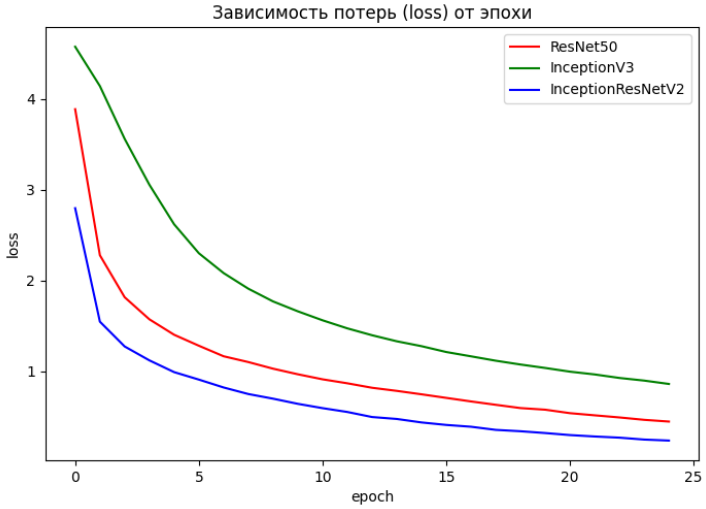


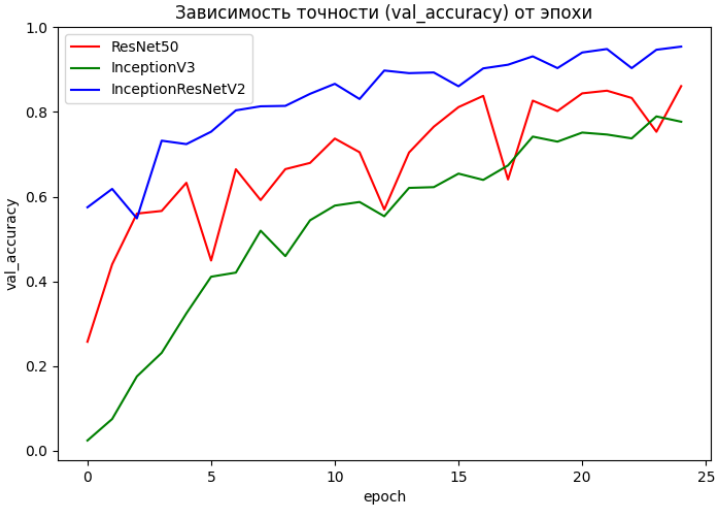
Конец результата запуска программы neurotest.py в режиме train

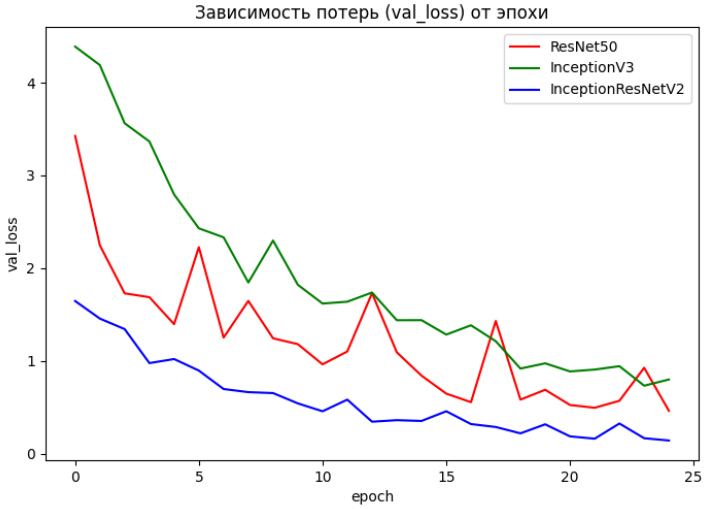
```
Epoch 00025: val_loss did not improve from 0.36723
Epoch 26/30
3156/3156 [==============================] - 1329s 421ms/step - loss: 0.3532 - accuracy: 0.8917 - val_loss:

Epoch 00026: val_loss improved from 0.36723 to 0.32274, saving model to D:\Apps\tf\food/models/ResNet50\foo
-0.3227-val_accuracy-0.9004.hdf5
Epoch 27/30
3156/3156 [==============================] - 1337s 423ms/step - loss: 0.3399 - accuracy: 0.8949 - val_loss:

Epoch 00027: val_loss did not improve from 0.32274
Epoch 28/30
3156/3156 [==============================] - 1347s 427ms/step - loss: 0.3240 - accuracy: 0.8993 - val_loss:

Epoch 00028: val_loss improved from 0.32274 to 0.26179, saving model to D:\Apps\tf\food/models/ResNet50\foo
-0.2618-val_accuracy-0.9184.hdf5
Epoch 29/30
3156/3156 [==============================] - 1332s 422ms/step - loss: 0.3104 - accuracy: 0.9039 - val_loss:

Epoch 00029: val_loss did not improve from 0.26179
Epoch 30/30
3156/3156 [==============================] - 1330s 422ms/step - loss: 0.2988 - accuracy: 0.9066 - val_loss:

Epoch 00030: val_loss improved from 0.26179 to 0.20615, saving model to D:\Apps\tf\food/models/ResNet50\foo
-0.2061-val_accuracy-0.9351.hdf5
```

Результаты работы neurotest.py









Средства мобильной разработки

Кросс-платформенные

- Flutter – Dart
- Unity – C++/C#
- Xamarin – C#
- Тонкие клиенты и их развитие – HTML5/React Native

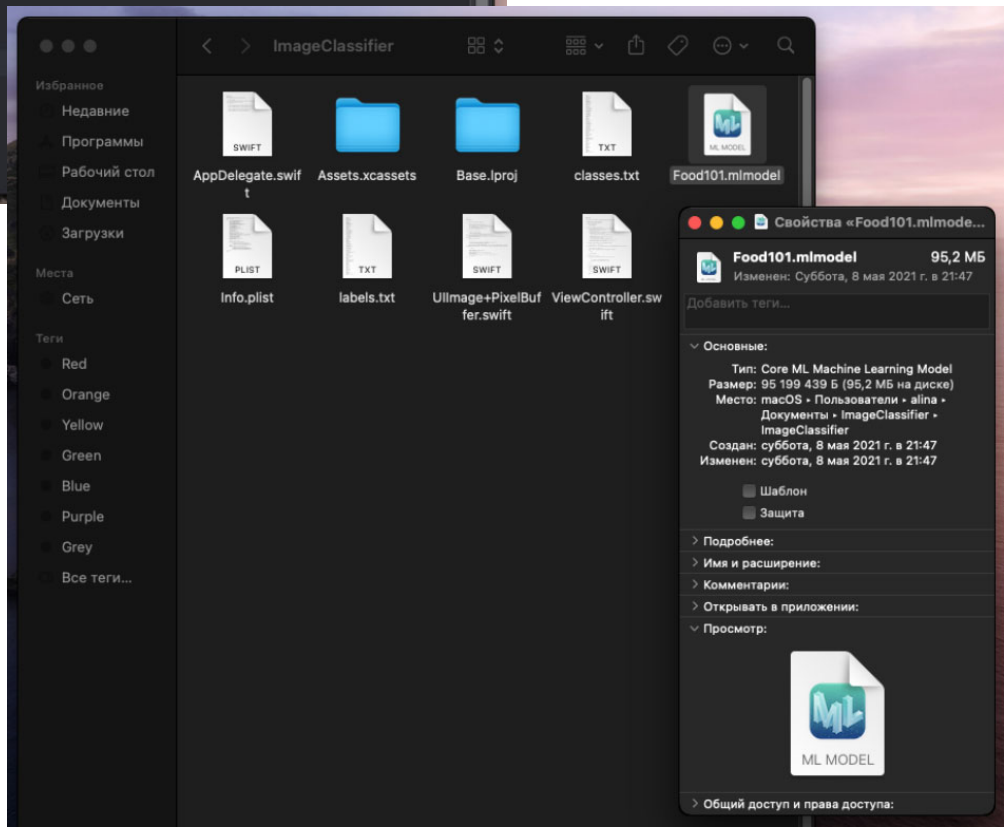
Нативные

- Android – Java/Kotlin
- iOS – Objective-C/Swift

Конвертирование Python модели в формат Core ML (food101.py)

food101.py 〉 No Selection

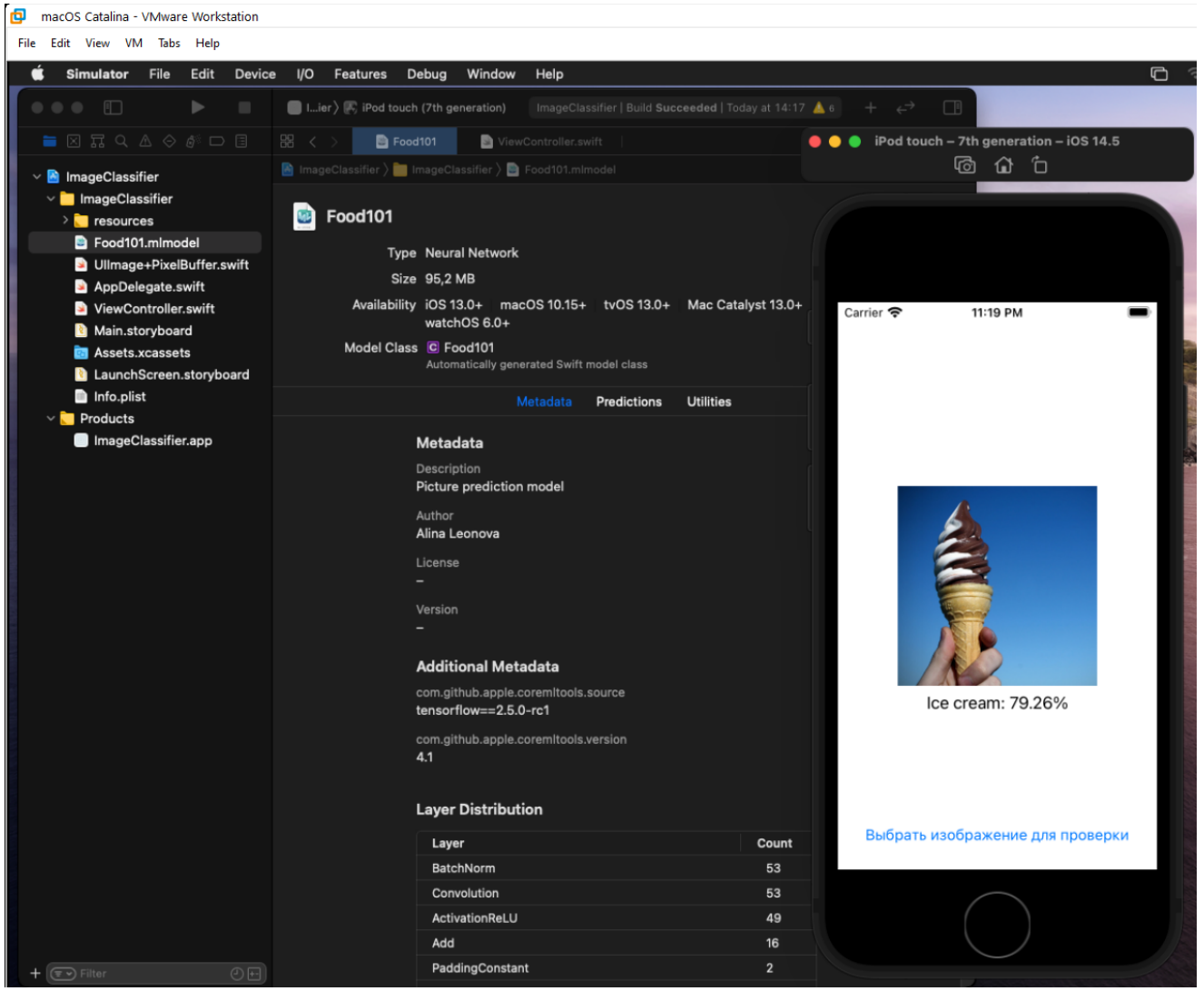
```python
import coremltools

coreml_model = coremltools.converters.keras.convert('./model4b.10-0.68.hdf5', input_names=['image'],
    output_names=['foodConfidence'], class_labels='labels.txt', image_input_names=['image'], image_scale=2/255.0,
    red_bias=-1, green_bias=-1, blue_bias=-1)
coreml_model.short_description = 'This model takes a picture of a food and predicts its name'
coreml_model.input_description['image'] = 'Image of a food'
coreml_model.output_description['foodConfidence'] = 'Confidence and label of predicted food'
coreml_model.output_description['classLabel'] = 'Label of predicted food'
coreml_model.save('Food101.mlmodel')

```



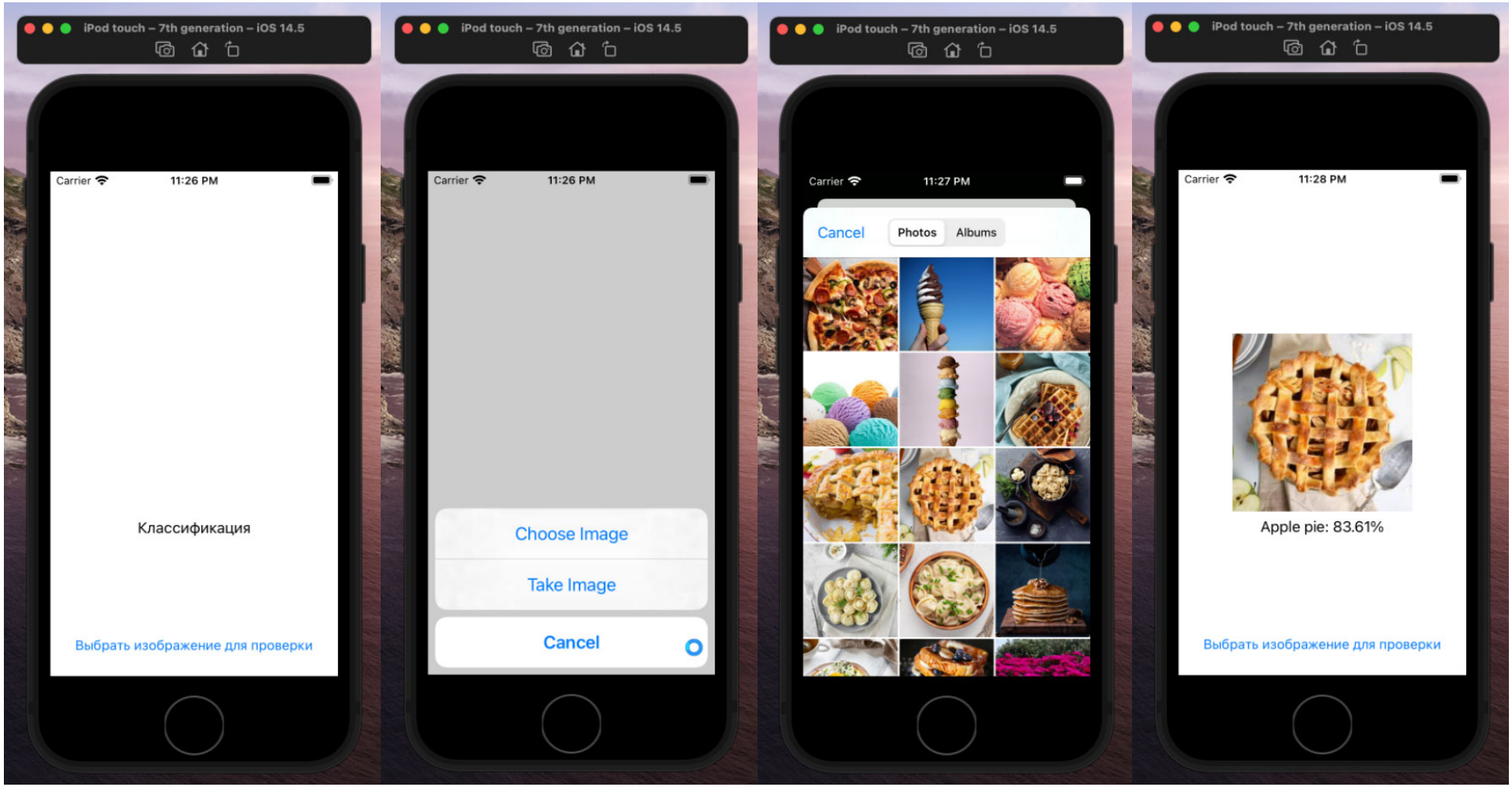
Функция processImage в файле ViewController.swift

```swift
func processImage(_ image: UIImage)
{
    let model = Food101()
    let size = CGSize(width: 256, height: 256)

    guard let buffer = image.resize(to: size)?.pixelBuffer()
    else
    {
        fatalError("Ошибка конвертации изображения!")
    }

    guard let result = try? model.prediction(input_1: buffer)
    else
    {
        fatalError("Ошибка классификации!")
    }

    var max:Float = -100
    var maxi:Int = 0
    for i in 0..<Int(result.Identity.strides[0])
    {
        if result.Identity[i].floatValue > max
        {
            max = result.Identity[i].floatValue
            maxi = i
        }
    }
    let maxs = String(format: "%.2f", max * 100.0)
    percentage.text = "\(getLabel(maxi)): \(maxs)%"
    imageView.image = image
}
```

Проект приложения в Xcode и его работа на симуляторе



Другие примеры работы приложения распознавания изображений еды на симуляторе



Результаты

- ✓ по итогам рассмотрения популярных библиотек Python, реализующих классические алгоритмы машинного обучения, был выбран и реализован вариант использования Keras поверх TensorFlow;
- ✓ на основе созданных и обученных моделей проведено сравнение различных архитектур нейронных сетей;
- ✓ рассмотрены фреймворки компании Apple для работы с технологиями машинного обучения и проведён перенос лучшей модели машинного обучения в формат Core ML;
- ✓ разработан прототип мобильного приложения для платформы iOS, классифицирующего еду на изображениях, и определены возможные направления его развития;
- ✓ основные результаты работы докладывались на конференции ITTMM’2021 и опубликованы в материалах конференции, а также была подана заявка на государственную регистрацию программы для анализа эффективности архитектур нейронных сетей на основе построенных моделей машинного обучения на заданном наборе изображений NeuroTestAL.

Спасибо за внимание