Tampereen ammattikorkeakoulu

Machine Learning and Mobile Development

Joonas Lauhala

BACHELOR'S THESIS
September 2020

Business Information Systems
Software Development

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn tutkinto-ohjelma
Ohjelmistotuotanto

LAUHALA, JOONAS:
Machine Learning and Mobile Development

Opinnäytetyö 36 sivua, joista liitteitä 1 sivu
Syyskuu 2020

Viime vuosikymmenen aikana koneoppimisen suosio on kasvanut räjähdysmäisesti. Mahdollisia käyttötarkoituksia koneoppimiselle tutkitaan jatkuvasti monella eri tieteenalalla ja tämän vuoksi koneoppimista hyödyntävän sovelluskehityksen perustarpeet kasvavat päivittäin.

Opinnäytetyön toimeksiantajana toimi Piceasoft Oy, joka kehittää sovelluksia mobiililaitteiden elinkaaren hallintaan. Piceasoftin tuoteportfolioon kuuluu esimerkiksi Diagnostics, Verify, Switch, Report, Eraser ja Trade-In. Osana Piceasoftin jatkuvaa tutkimus- ja -kehitysprosessia kokeillaan koneoppimista hyödyntäviä sovellusratkaisuja.

Tämän opinnäytetyön tarkoituksena on antaa perustiedot koneoppimista käyttävän mobiilisovelluksen kehityksestä. Opinnäytetyö haki vastausta seuraavanlaisiin kysymyksiin: ”Kuinka paljon sovelluskehittäjän täytyy tietää koneoppimisesta, jotta voi käyttää sen etuja omissa sovelluksissa?”, ja ”Minkä seikkojen vuoksi koneoppimis-sovelluskehyksen käyttö on suosittua?”

Opinnäytetyön tuloksena kehitettiin koneoppimis-sovelluskehystä käyttävä mobiilisovellus Android- ja iOS-alustoille. Mobiilisovellus luokittelee käyttäjän piirtämiä numeroita valmista koneoppimismallia käyttäen.

Keskeisimpänä havaintona todettiin, että koneoppis-sovelluskehyksen käyttö on tehty helpoksi, ja tämän vuoksi perustiedot koneoppimisesta riittävät hyvin yksinkertaisten mobiilisovellusten luomista varten. Lisäksi työn aikana huomattiin, että suurinta osaa koneoppimiseen liittyvästä opetusmateriaalista ei ole kirjoitettu sovelluskehittäjää ajatellen.

Asiasanat: koneoppiminen mobiili sovelluskehitys teknologia

ABSTRACT

Tampere University of Applied Sciences
Degree Programme in Business Information Systems
Option of Software Development

LAUHALA, JOONAS:
Machine Learning and Mobile Development

Bachelor's thesis 36 pages, appendices 1 page
September 2020

The popularity of machine learning technologies has increased exponentially in the past ten years. In many fields of study, possible use cases for machine learning are tested continuously. Thus, the basics of developing software that uses machine learning are becoming more critical by the day.

The client of this Bachelors' thesis was Piceasoft LLC. Piceasoft is a company, who develops software to manage the mobile device lifecycle. Piceasoft has multiple products such as Diagnostics, Verify, Switch, Report, Eraser, and Trade-In. Machine learning is being tested as a part of the continuous application research and development process.

The purpose of this thesis was to increase knowledge of how to develop basic machine learning applications for mobile platforms. The thesis sought answer to questions: "As a developer, how much do you need to know about machine learning to start using it in your applications?" and "Why is the development with a machine learning framework popular?"

A mobile application for Android and iOS platforms that used a machine learning framework was developed as a result of this thesis. The application classifies handwritten digits by using an existing machine learning model.

The most significant finding of this thesis was that development with a machine learning framework is simple. The basics of machine learning are often good enough to start the mobile application development. Also, it was noted that most of the existing learning resources for machine learning are not written for software developers.

Keywords: machine learning mobile software development technology

TABLE OF CONTENTS

ABBREVIATIONS AND TERMS

API

Abbreviation for Application Programming Interface. API is an interface that defines interactions between multiple software intermediaries.

APK

Abbreviation for Android Package. APK is a package that contains all Android application resources necessary for the distribution.

Dataset

Collection of separate sets of information, being used as a single unit.

IDE

Abbreviation for Integrated Development Environment. IDE is a software application that contains all the necessary tools to develop, execute, and build software.

Inference function

The real-world equivalent of the reasoning process, but for the computer.

Model

The real-world equivalent of storing past experiences, but for the computer.

SDK

Abbreviation for Software Development Kit. A collection of software development tools to ease the development process for a specific platform.

Software platform

Significant piece of software like an operating system (Windows, Linux, Mac OS, iOS, or Android) that hosts smaller applications designed to run in it.

Software framework

Abstraction, in which software providing generic functionality can be selectively extended by additional user-code.

Android

Linux-based operating system used mostly for mobile devices, such as phones and tablets, maintained by Google.

iOS

Operating system used for mobile devices manufactured by Apple.

1 INTRODUCTION

1.1 Background

The popularity of machine learning has increased exponentially in the past ten years. In many fields of study such as Computer science, Engineering and Technology, Economics, and Medicine, possible use cases for machine learning are tested continuously. Thus, the basics of developing software that uses machine learning are becoming more critical by the day. Machine learning has potential that reaches far into the future.

For the reasons mentioned above, this Bachelors' thesis aims to improve knowledge on how to develop basic machine learning applications for mobile platforms. The thesis only covers the basics since machine learning is such a complex topic, and it might be too much to comprehend. Machine learning also has a long history, and what has led to the current state of machine learning technology being popular is not essential.

The main goal of this thesis is to combine all the necessary information to get started with machine learning application development. Because software development and machine learning are tightly coupled, the secondary goal of the thesis is to act as a bridge between the complex world of machine learning and software development. Machine learning terminology requires some simplifications to make the transition more transparent to the developers. The core concepts of machine learning technology are explained in a simplified form.

Piceasoft LLC acts as a client of this Bachelors' thesis. Piceasoft develops software solutions to manage the mobile device lifecycle. As more machine learning-centric products are planned, a primer for the underlying technology can be useful for most software developers.

1.2 Client

Founded in 2012 at Tampere, Finland, Piceasoft LLC is a company that develops software and applications to manage the mobile device lifecycle. Piceasoft has multiple products such as Diagnostics, Verify, Switch, Report, Eraser, and Trade-In. These products offer ways to ease the recycling process and can be used to solve common issues when reselling mobile devices. Piceasoft customers are recycling companies, resellers, and telephone operators.

1.3 Goal

The first goal of this thesis is to create an instruction manual of sorts by gathering everything essential for developing applications, which use machine learning and available frameworks, including information about machine learning methods, models, general applications, techniques, suitability, and more.

Secondly, to fully understand the basic concepts of machine learning development, a “handwritten digit classifier,” demonstrating the development process, is an essential factor. The project uses a machine learning framework, which allows the development process while also showcasing the required setup and dependencies when using such a framework. The demo should allow the user to draw digits by hand and submit them to classification with a press of a button.

The mobile demo application should be implemented on major platforms, in this case, Android and iOS. The application architecture should remain similar among both platforms, and the same core functionality should be exposed. The application should also use the same machine learning model in both implementations.

The thesis seeks an answer to questions, such as: “As a developer, how much do you need to know about machine learning to start using it in your applications?”, “What is the relationship between machine learning and traditional software development?” “Are there any major differences when using a machine learning framework on different platforms?” and “Why is development with a machine learning framework popular?”

2 MACHINE LEARNING EXPLAINED

2.1 What is Machine Learning?

The term “machine learning” is quite self-explanatory. In simple terms, machine learning (or “ML” for short) is a capability for the computer to start making decisions without being explicitly instructed each time (Ng 2018). Machine learning is a sub-field of artificial intelligence (or “AI”).

Gopalakrishnan & Venkateswarlu (2018) state that when a computer program can improve the way that the tasks are performed by using its previous experience, we can then say that the computer is learning. This scenario is entirely different from one where a program performs a task based on existing rules, logic, and information added by a programmer. For example, a chess game where the programmer implements a winning strategy for the program, illustrates this perfectly. The machine learning approach differs from that because it does not have any clue on how to win the game at first. The programmer only defines a set of legal moves, and the rest is for the computer to learn. By learning from its mistakes, the computer starts to win eventually, but this requires plenty of repetition. (Gopalakrishnan & Venkateswarlu 2018.)

Machine learning is an iterative process for the computer (Figure 1), meaning that it takes some time to complete (Gopalakrishnan & Venkateswarlu 2018), just like the process for humans to learn to ride a bicycle. The process requires much previous experience, trial, and error to achieve mastery in a specific task.

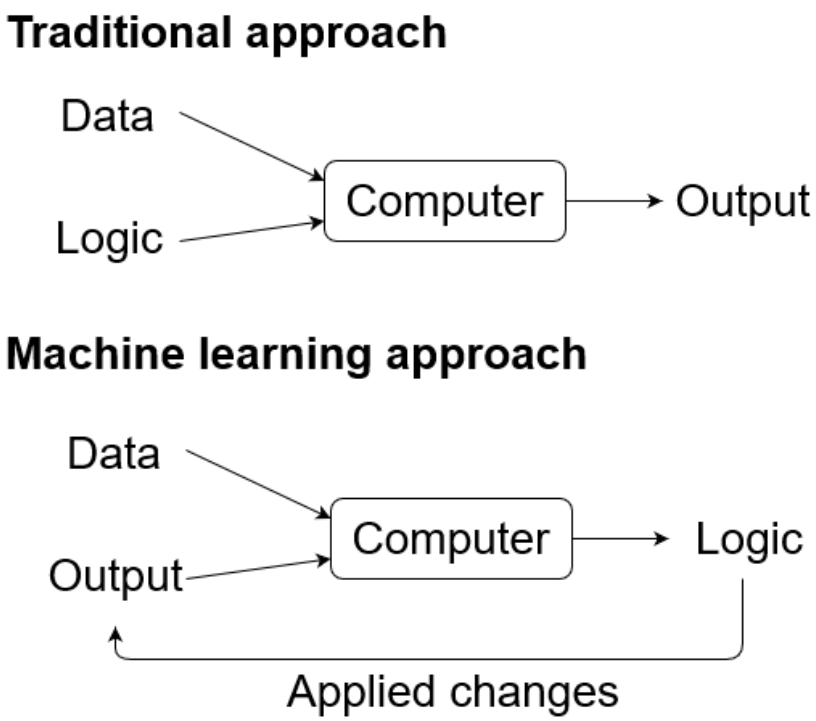

FIGURE 1. Difference between a traditional and a machine learning approach

Machine learning simplifies many everyday tasks that would be tedious for a programmer to code traditionally. Typical tasks that would otherwise require infinite lines of code because they do not have a well-defined set of rules, meaning that they are non-existent or too abstract for the programmer to follow. The information processing also takes time, and in real-time applications, a way for the computer to make assumptions based on previous experiences will reduce the processing time significantly. (Gopalakrishnan & Venkateswarlu 2018.)

Computers were the first ones to use machine learning, but it is proven that mobile development with machine learning implemented on mobile devices is trendy. Modern mobile devices show a high productive capacity level that is close enough to perform appropriate tasks to the degree that modern computers do. (Gopalakrishnan & Venkateswarlu 2018.)

Generally speaking, any application that does one of the following is using machine learning in some way:

- Speech recognition
- Computer vision and image classification

- Gesture recognition
- Translation from one language into another
- Interactive on-device detection of text
- Autonomous vehicles, drone navigation, and robotics
- Patient-monitoring systems and mobile applications interacting with medical devices

(Gopalakrishnan & Venkateswarlu 2018.)

2.2 Basic requirements: Learning methods

Machine learning can, on a larger scale, be categorized into supervised- and unsupervised learning methods (Ng 2018). Other methods, like semi-supervised and reinforced learning (Figure 2), are out-of-scope of this thesis.

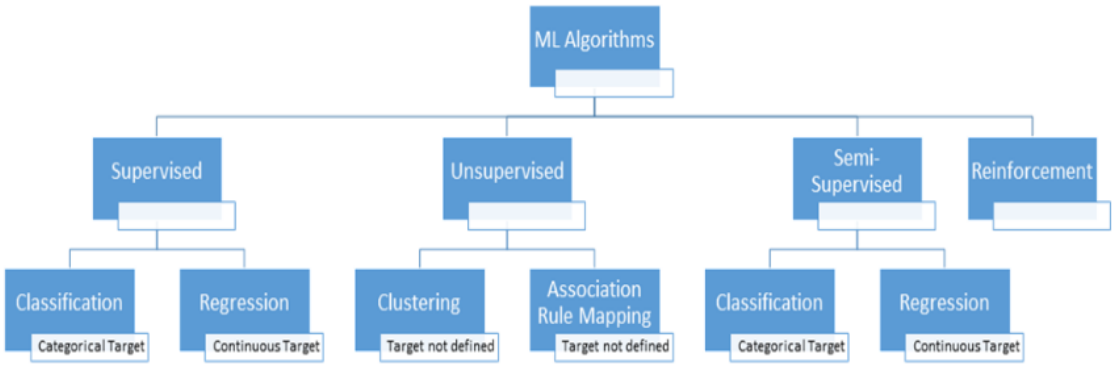

FIGURE 2. Different learning methods (Gopalakrishnan & Venkateswarlu 2018)

2.2.1 Supervised learning

The main goal of supervised learning is to create a function that maps inputs to outputs so well that the unforeseen input (x) can be mapped to output (y) by predicting where it should belong based on prior knowledge (Ng 2018). A set of single-digit numbers is a good example that is also used in the demo project later. When all ten digits are labeled, and a new image of a number comes in, the model should be able to predict what number it was (Figure 3).

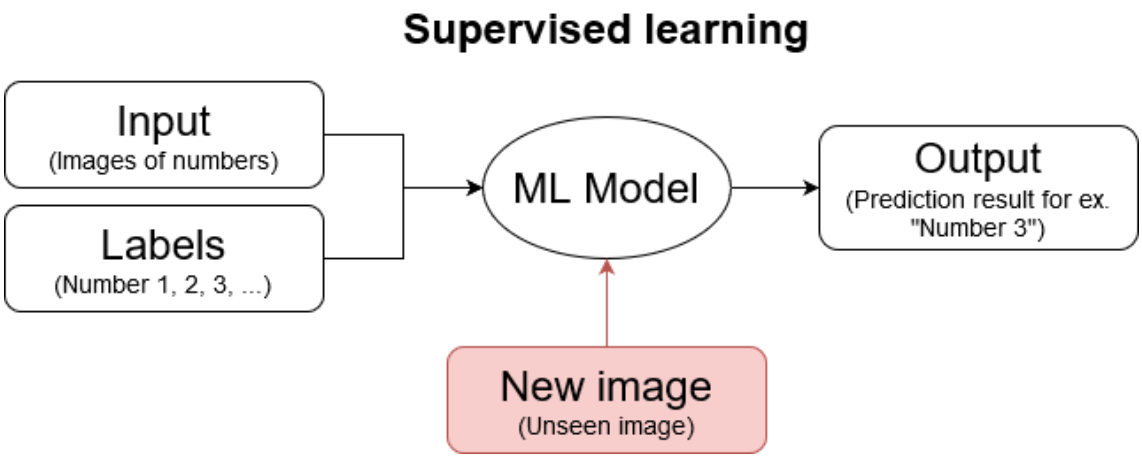

FIGURE 3. Supervised learning method (Loukas 2020, modified)

In supervised learning, if the predicted output is a discrete value, such algorithms are part of classification algorithms. However, when the predicted output is a continuous value, such algorithms fall under the umbrella of regression algorithms. These differences must be taken into consideration when training the model. (Gopalakrishnan & Venkateswarlu 2018.)

Gopalakrishnan & Venkateswarlu (2018) lists multiple algorithms that are associated with the supervised learning, such as K-nearest neighbours, Naive Bayes, Decision trees, Linear regression, Logistic regression, Support vector machines, and Random forest. (Gopalakrishnan & Venkateswarlu 2018.)

2.2.2 Unsupervised learning

On the contrary, unsupervised learning does not learn under supervision. The model learns based on the data that gets fed to it and discovers hidden patterns in the data (Figure 4). This type is useful when there are enormous amounts of data, and the patterns we are looking for are unknown. Unsupervised learning algorithms can, in general, provide useful insights about given data like confirm what we might already know or, in some cases, predict what is going to happen next. A suitable example of using unsupervised learning could be customer segmentation. When vast amounts of customer data are available that captures all customer features, unsupervised learning algorithms could cluster different kinds

of customers. This type of information could be precious to the marketing team. (Gopalakrishnan & Venkateswarlu 2018.)

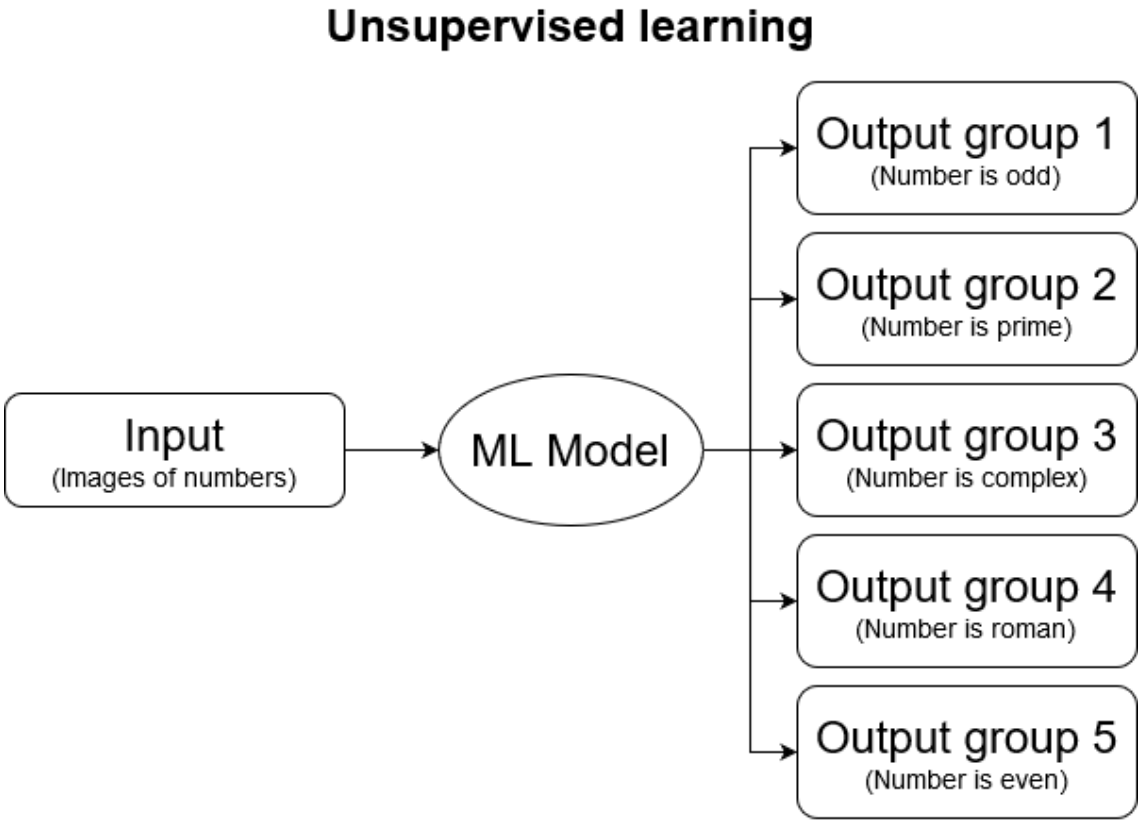

FIGURE 4. Unsupervised learning method (Loukas 2020, modified)

According to Gopalakrishnan & Venkateswarlu (2018), the following algorithms belong to unsupervised learning. Clustering algorithms such as Centroid-based algorithms, Connectivity-based algorithms, Density-based algorithms, Probabilistic, Dimensionality reduction, Neural networks/deep learning, and finally, Association rule learning algorithm. (Gopalakrishnan & Venkateswarlu 2018.)

2.3 Basic requirements: Models and training

Machine learning uses models to store past experiences; the models are then used by the computer to apply the machine learning algorithm to the problem in hand. Model formats vary between different machine learning frameworks. As a side-effect, this causes most of the existing model formats to be incompatible with

each other. A particular script or program is needed to convert between different model formats. The model storage space requirements vary greatly too, but usually, the models optimized for mobile platforms are much smaller than their desktop counterparts.

Model training can be done from scratch and can be a tedious, time-consuming task. Training from scratch can, however, yield more reliable results if done correctly. One of the prerequisites is that the problem needs to be well-defined. When the machine learning problem is well-defined, the following conditions are satisfied: We have the right problem, the right data, and the right criteria for success. Sometimes this is the only way to approach a particular issue because application data requirements can be precise (Gopalakrishnan & Venkateswarlu 2018).

An alternative way is to use an existing model if a suitable model is available and retrain it based on application needs. Using the existing model can save valuable time and resources at the possible cost of model accuracy. That is why the model used for the basis should be selected very carefully (Gopalakrishnan & Venkateswarlu 2018).

In both cases, the model requires a useful and extensive dataset, meaning that it is qualitative but also quantitative. In the end, when training the model, it is all about following the best practices available. On the list below, there are a few examples of these best practices.

- Use multiple success metrics
- Use metrics that are easy to understand
- Use metrics that make the model comparisons easier
- Introduce randomness to training data in order to avoid biases
- Reduce useless or redundant training data

2.4 On-device Machine Learning

Ng (2018) says that nowadays, people are spending more time with mobile devices, and it makes much sense to run machine learning models on the device itself instead of storing it in the cloud (Ng 2018).

The most significant advantage of the on-device model is that model is always available without the need to send information back and forth (Figure 5). Localness is also the reason why using the model is convenient.

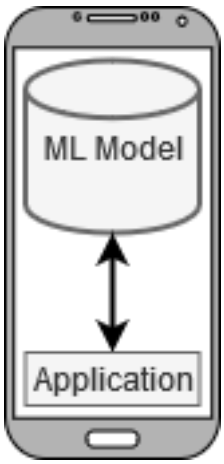

FIGURE 5. Relationship between mobile application and local (on-device) model

However, the Cloud model is not useless because the Cloud model can be updated seamlessly, without users ever noticing the difference. The Cloud model is the way to go if the model file is enormous and would drastically increase the application size (Figure 6).

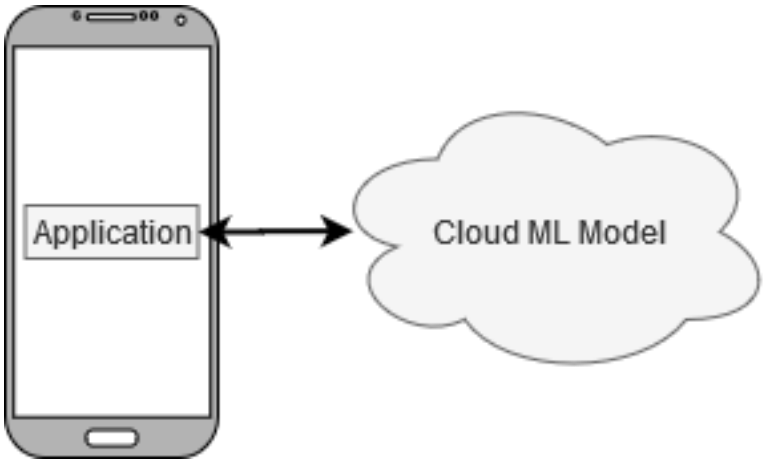

FIGURE 6: Relationship between the mobile application and the cloud model

TABLE 1. On-device vs. Cloud model (Gopalakrishnan & Venkateswarlu 2018)

On-device model	**Cloud model**
Locally stored	Stored in Cloud service
Fast: Almost instantaneous to use	Slow: uses requests and responses
Tedious to update quickly for all users	Seamless updates to all users
Increases application size	Does not affect application size
Platform portability might cause issues	Portable between different platforms
Data can be kept private	Data can be shared with the third party

Training models on a device has still not risen to popularity, since the computing power and storage space requirements are quite large (Ng 2018). Mobile is not the preferred platform to train models, the only exception being tiny datasets. The retraining phase would also become too complicated. (Gopalakrishnan & Venkateswarlu 2018.). However, Core ML 3 for iOS allows the developer to leverage this functionality for simple tasks.

3 POPULAR MACHINE LEARNING FRAMEWORKS FOR MOBILE PLATFORM

3.1 Google: Tensorflow Lite

Tensorflow Lite by Google is an open-sourced cross-platform machine learning framework aimed at mobile development. Tensorflow Lite is the lightweight version of its big brother, Tensorflow, and is suitable for microcontrollers, mobile, and embedded devices (such as the Internet of Things). (Khandelwal 2020.) The newest stable release of Tensorflow Lite is version 2.2.0.

Tensorflow Lite is a popular machine learning framework primarily used in Android applications. Newer versions of Tensorflow Lite can use Android Neural Network API (NNAPI) to accelerate the inference functionality. The framework supports a wide variety of different models, but the most common model format is TFLite-format.

Supported mobile platforms

- Android
- iOS

The usage of Tensorflow Lite on Android and iOS platforms is very straightforward. Both platforms require the framework as a dependency. The only real difference is that on iOS, the project must use a CocoaPods dependency manager.

3.2 Apple: Core ML

Core ML is a machine learning framework from Apple. First introduced in the WWDC 2017 conference, Core ML is suitable for developing machine learning applications for the iOS platform, includes the support for image analysis, sound

analysis, language recognition, and audio-to-text conversion (Wiggers 2018.) The newest stable release of the Core ML framework is Core ML 3.

Core ML is a popular machine learning framework mainly used on Apple platforms. The framework takes advantage of the hardware components of Apple devices. Core ML supports only MLModel-format (".mlmodel"), but various popular model formats are convertible to MLModel.

Supported mobile platforms

- iOS

iOS SDK contains the Core ML framework by default. The developer must import the API within code before its use.

3.3 Google: ML Kit for Firebase

Firebase ML Kit (or ML Kit for Firebase) is a bundle of machine learning frameworks by Google. It launched in May 2018. ML Kit SDK packages Google Cloud Vision API, Tensorflow Lite, and Android Neural Networks API together. ML Kit can be used to detect faces, scan barcodes, label images, and for other useful purposes. It supports on-device data models along with cloud data models. (Wiggers 2018.)

ML Kit for Firebase is an easy way to use machine learning for common problems, such as reading the contents of a QR code or detecting and reading a text from an image. A feature of ML Kit that is not common knowledge allows the framework to load and use custom Tensorflow Lite models. However, because ML Kit API is more nuanced towards everyday use cases – it does not allow as much control over the model as Tensorflow Lite does.

Supported mobile platforms

- Android

- iOS

Firebase ML Kit can be used on Android and iOS platforms to a varying degree. The Android version of the framework requires only a few project dependencies to function correctly. Some additional steps must be taken when using the ML Kit on the iOS platform. For example, the iOS project must use a CocoaPods dependency manager, a few API functionalities require a newer version of Xcode IDE, and some parts of the framework differ between platforms.

4 APPLICATIONS FOR MACHINE LEARNING

4.1 General applications

Various features used in existing applications use machine learning. High-level examples of these features are email spam identification systems, product recommendations on e-commerce sites, and automatic face tagging functionality on social media. All formerly mentioned features use machine learning to a varying extent. (Ng 2018.)

Today, many applications rely heavily on using machine learning frameworks for specific tasks, but still use traditional solutions for most of their functionality. Machine learning is not a replacement for conventional ways of developing software but an extension of it. In many cases, the machine learning approach can simplify the convoluted logic of specific tasks.

According to Gopalakrishnan & Venkateswarlu (2018), these popular mobile applications use machine learning.

- Google Maps
- Facebook
- Snapchat
- Netflix
- Tinder
- Uber

(Gopalakrishnan & Venkateswarlu 2018.)

There are many other popular applications not listed that use machine learning functionality. What comes to using machine learning functionality in viral applications is a smart move because, firstly, it can massively improve the user experience and, secondly, provide information with actual business value to the developer(s) or the company.

4.2 Image classification

Image classification is arguably the best-known computer vision technique. The image classification works by categorizing the images based on their most dominant content (Image 1), and supervised learning algorithms are crucial when training for this kind of task. The image classification technique is the basis for the other computer vision techniques and is a critical part when the task has something to do with images.

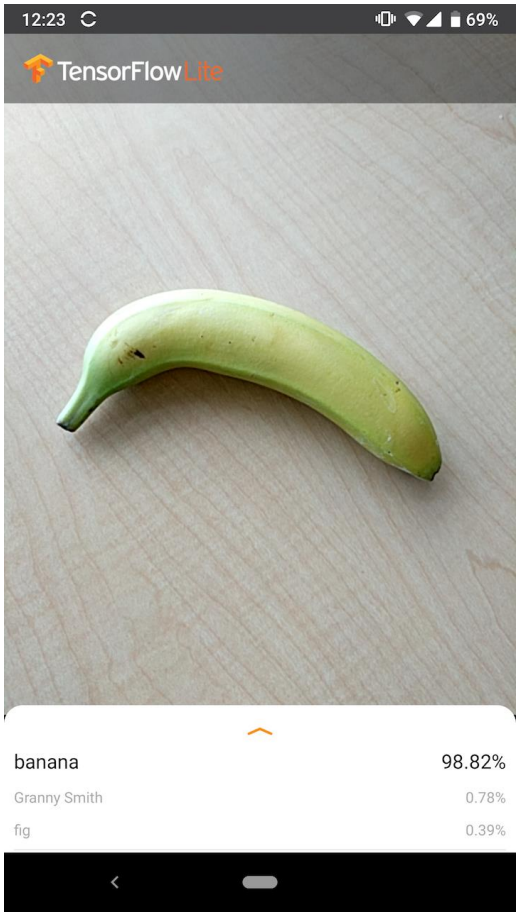

IMAGE 1. Image classification of “a banana” with confidence percentage in Tensorflow Lite demo app (Image classification | Tensorflow Lite 2020)

A mobile application demo that is implemented during the last chapters of this thesis uses the image classification technique to categorize handwritten digits to their corresponding numbers.

4.3 Object detection

Object detection is the second computer vision technique briefly covered in this thesis. It allows the detection of objects within an image or video feed.

Machine learning can be used to detect objects in images (Image 2). For example, pictures that contain devices, toys, furniture, and other items. Object detection also works as a prerequisite for manipulating images based on their content, like swapping face with another.

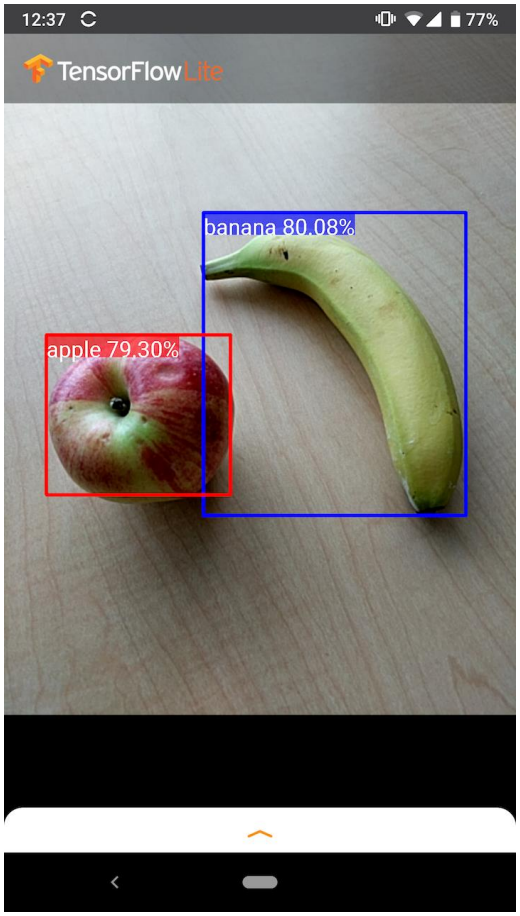

IMAGE 2. Bounding box and confidence percentage of “an apple” and “a banana” in Tensorflow Lite demo app (Object detection | Tensorflow Lite 2020)

5 USING MACHINE LEARNING FRAMEWORK IN A PROJECT

5.1 Foreword and project description

The machine learning framework used in a mobile application project is to showcase the basic functionality and all the necessary details for the development process. All project-related source code is available for viewing/downloading on GitHub (Appendix 1).

Application development with machine learning frameworks does not require in-depth knowledge of machine learning algorithms, model creation/training, or other intricacies. Only basic mobile application development skills are necessary. Developers need to know how to use provided API, import model(s), and go from there. (Gopalakrishnan & Venkateswarlu 2018.)

This demo application classifies digits as they are drawn to the canvas by the user and tries to predict the correct number. Both platforms use the same MNIST model for digit classification. The project (or thesis) does not explain how to choose the best framework for a specific problem, and it is a discussion of its own. The most popular and preferable frameworks for the platforms were selected.

1. User must be able to draw digits on the screen
2. The application must be able to classify drawings with machine learning
3. Predicted digit and confidence score should be visible to the user
4. User must be able to clear his/her drawing and start over

5.2 Project dependencies and setup

“Thesis-project” uses the empty template as the basis on both projects. The first step when setting up the project was to modify the layout of the application that is visible to the end-user. The application layout was kept similar on both platforms for consistency.

An MNIST data set is a subset of NIST, containing a training set of 60 000 and a test set of 10 000 examples of handwritten digits by 500 different writers. The numbers are centered, size-normalized, and fixed to 28x28 pixels per image. Original images from the NIST set are black and white. (LeCun & Cortes & Burges 2013.) The MNIST based machine learning model was used on both platforms.

5.3 The development process in detail

5.3.1 Android-platform

The Android demo application was developed by using the Android Studio IDE (Image 3). The Android implementation uses the TensorFlow Lite library as its machine learning framework.

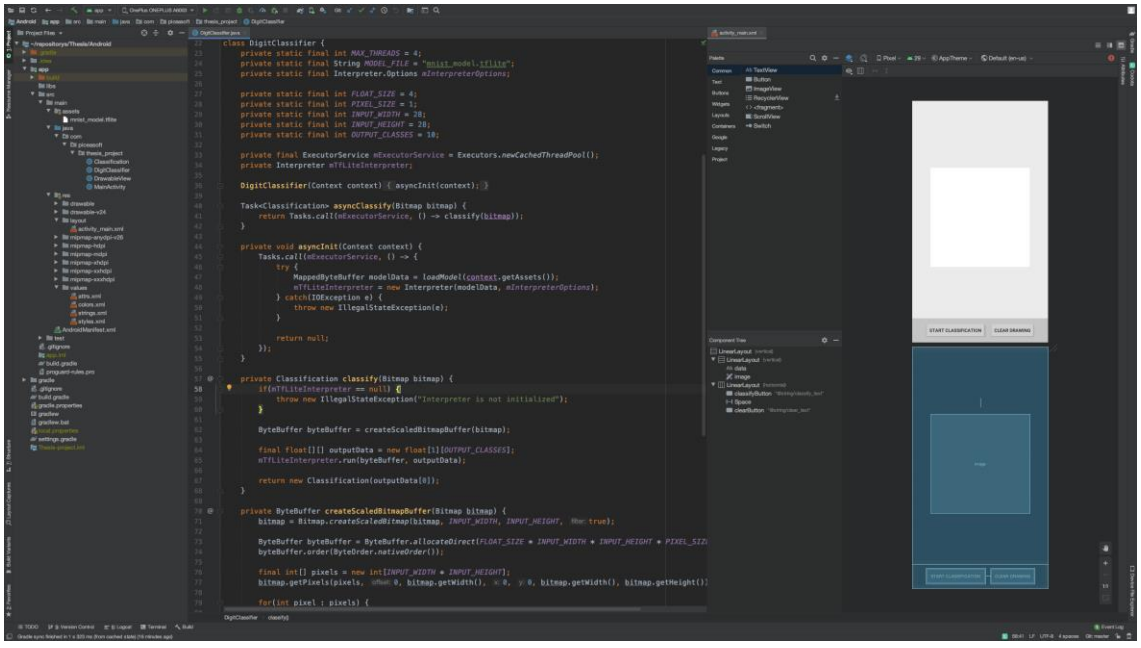

IMAGE 3. Project structure and application layout in Android Studio IDE

The first steps of development on the Android-platform were to create a new project with Android Studio from an empty template and define application layout, along with adding the following dependencies to the *app*-level Gradle-file. The Android implementation uses the MNIST in a TFLite format (“*.tflite”).

TABLE 2. The most significant dependencies of the Android project

Dependency name	**Dependency source**
Tensorflow Lite	org.tensorflow:tensorflow-lite:2.1.0
Tasks	com.google.android.gms:play-services-tasks:17.1.0

After including project dependencies, among other things, a class hierarchy was set. The Android project contains a total of four classes.

TABLE 3. Classes of the Android project

Class name	**Description**
MainActivity	The main class of the application (auto-generated)
DrawableView	Component class allowing the user to draw into it
DigitClassifier	Loads and uses the TFLite model to classify input
Classification	Stores predicted digit and confidence

The main activity (MainActivity) opens when the user launches the application. The main activity loads the application layout and handles user interactions. While constructing the main activity, a digit classifier (DigitClassifier) will load and initialize the TFLite-formatted MNIST model in the background (Figure 7). A reference to the digit classifier (DigitClassifier) is stored in a variable within the main activity class. Because Android SDK does not provide a built-in view component to draw lines to the screen, a custom-made drawable view (DrawableView) is needed to accomplish this task.

```
DigitClassifier(Context context) {
    asyncInit(context);
}

private void asyncInit(Context context) {
    Tasks.call(mExecutorService, () -> {
        try {
            MappedByteBuffer modelData = loadModel(context.getAssets());
            mTfLiteInterpreter = new Interpreter(modelData, mInterpreterOp-
tions);
        } catch(IOException e) {
            throw new IllegalStateException(e);
        }

        return null;
    });
}
```

FIGURE 7. Sample code of ML model initialization

Each visible component on the application responds to user touch events. For example, the drawable view component responds to the user given touch coordinates and draws lines inside the view based on those. Under the hood, all buttons have a click listener (Figure 8), meaning that after the user has pressed the button, a callback function tied to that specific button gets executed.

```
public void buttonClickListener(View view) {
    switch(view.getId()) {
        case R.id.clearButton: {
            mDrawableView.reset();
            updateText(null);

            break;
        }
        case R.id.classifyButton: {
            mDigitClassifier.asyncClassify(mDrawableView.getRenderView())
                            .addOnSuccessListener(this::classificationFinished)
                            .addOnFailureListener(this::classificationFailed);

            break;
        }
    }
}
```

FIGURE 8. Sample code of a button callback function

The classification process starts with a button click. A copy of the user drawn bitmap gets transformed to suit the MNIST model requirements. The bitmap then gets sent to the classifier, which uses the MNIST model inference function that outputs an array of confidence scores for each number. A data class (Classification) gets instantiated with the result array (Figure 9), and the number with the

highest confidence score is stored within the object along with its confidence score.

```
Task<Classification> asyncClassify(Bitmap bitmap) {
    return Tasks.call(mExecutorService, () -> classify(bitmap));
}

private Classification classify(Bitmap bitmap) {
    if(mTfLiteInterpreter == null) {
        throw new IllegalStateException("Interpreter is not initialized");
    }

    ByteBuffer byteBuffer = createScaledBitmapBuffer(bitmap);

    final float[][] outputData = new float[1][OUTPUT_CLASSES];
    mTfLiteInterpreter.run(byteBuffer, outputData);

    return new Classification(outputData[0]);
}
```

FIGURE 9. Sample code of image processing and classification

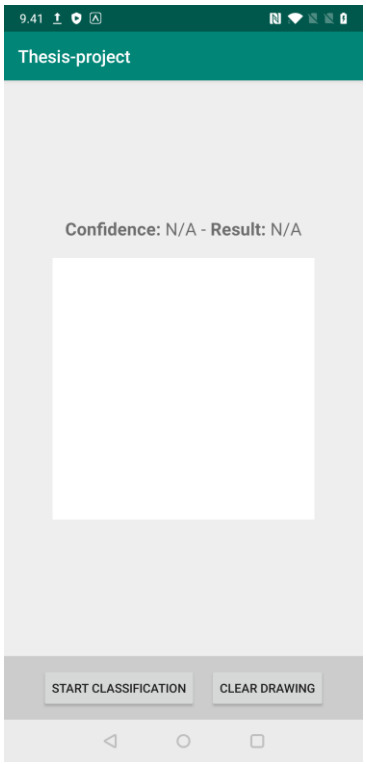

IMAGE 4. An initial view when launching the Android application

5.3.2 iOS-platform

The iOS demo application was developed by using the Xcode IDE (Image 5). The Core ML library is used as an iOS application machine learning framework.

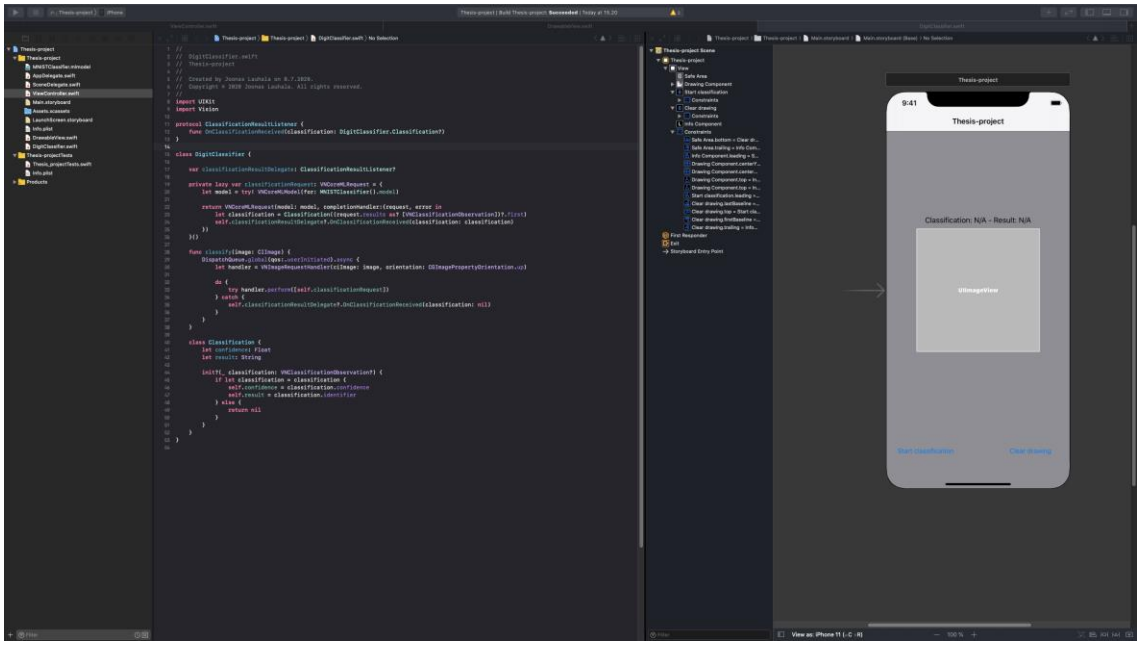

IMAGE 5. Project structure and application layout in Xcode IDE

iOS project development started by generating an empty project structure with Xcode. The iOS implementation uses the MNIST in an ML model format (“*.mlmodel”).

Vision (Core ML) is a built-in library and only needs import-statement on top of the class before use. Just like on Android-platform, a class hierarchy was set containing a total of five classes.

TABLE 4. Classes of the iOS project

Class name	**Description**
AppDelegate	Launcher class (auto-generated)
SceneDelegate	Application scene class (auto-generated)
ViewController	Main view controller (auto-generated but renamed)
DrawableView	Custom component for drawing
DigitClassifier	Loads and uses the ML model (in MLmodel format)

The AppDelegate class works as the entry point when launching the iOS application. The SceneDelegate loads the storyboard (or the layout of the iOS application) and hands the control to the ViewController class. The ViewController is responsible for handling user interaction. In the iOS project, the initialization of the MNIST model happens when the model gets used for the first time (Figure 10).

```
private lazy var classificationRequest: VNCoreMLRequest = {
    let model = try! VNCoreMLModel(for: MNISTClassifier().model)

    return VNCoreMLRequest(
        model: model,
        completionHandler: {request, error in
            let classification = Classification(
                (request.results as? [VNClassificationObservation])?.first
            )

            self.classificationResultDelegate?
                .OnClassificationReceived(classification: classification)
        }
    )
}()
```

FIGURE 10. Sample code of ML model initialization

All the visible components on the storyboard are linked to the ViewController. The standard iOS view components do not include a component that would allow the user to draw to the screen. Because of this, the implementation of a custom drawing component (DrawableView) is required. Each button on the storyboard has a callback function assigned to it, which gets called when the user presses the button (Figure 11).

```
@IBAction func onClassifyClicked(_ sender: Any) {
    if !drawingComponent.isEmptyCanvas() {
        if let image = drawingComponent.getCurrentImage() {
            digitClassifier.classify(
                image: image.applyingFilter("CIColorInvert")
            )
        }
    }
}

@IBAction func onClearClicked(_ sender: Any) {
    drawingComponent.image = nil
    OnClassificationReceived(classification: nil)
}
```

FIGURE 11. Sample code of button callback functions

A button press initiates the classification process, which in turn creates a new classification request with the current drawing (Figure 12). After the classification request has finished, the delegate class (Classification) will receive the results.

```
func classify(image: CIImage) {
    DispatchQueue.global(qos:.userInitiated).async {
        let handler = VNImageRequestHandler(
            ciImage: image,
            orientation: CGImagePropertyOrientation.up
        )

        do {
            try handler.perform([self.classificationRequest])
        } catch {
            self.classificationResultDelegate?
                .OnClassificationReceived(classification: nil)
        }
    }
}
```

FIGURE 12. Sample code of image classification

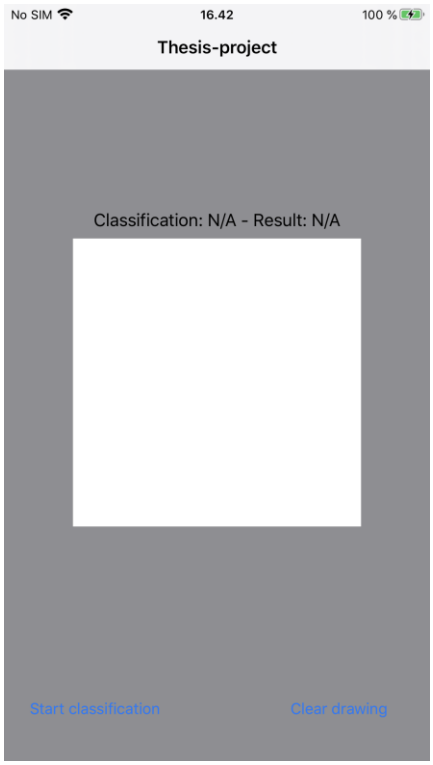

IMAGE 6. An initial view when launching the iOS application

5.4 Project summary

As can be seen from the simplicity of the application, creating a basic application that uses a machine learning framework is not complicated for a developer who already knows the basics of software development. The application allows users to draw digits and submit them for classification via a press of a button, as initially intended. The application displays the classification results to the user in the form of a text.

Both applications can classify user has drawn single-digit numbers correctly (Image 7 and Image 8), but for some unknown reason, the iOS application can classify most numbers a bit more reliably. A possible cause is a way that the drawings/images are handled; despite being very similar, the final implementations differ lightly.

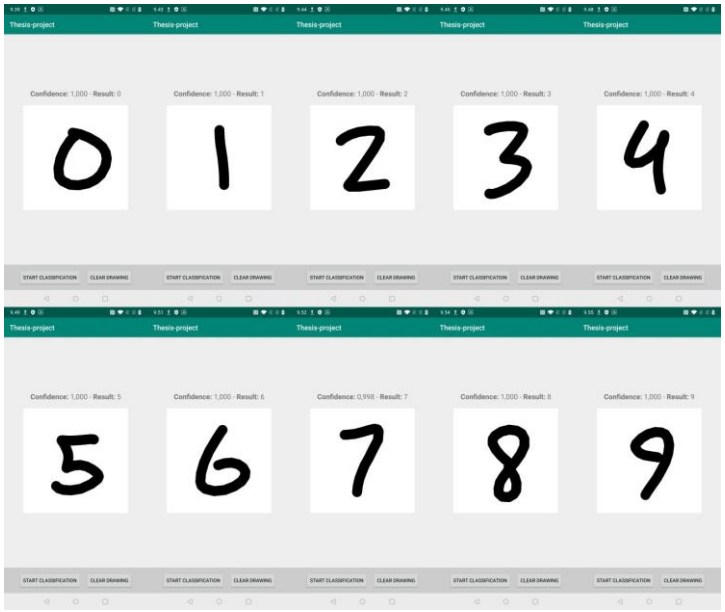

IMAGE 7. All single-digit numbers classified by the Android application

IMAGE 8. All single-digit numbers classified by the iOS application

In the case where the user draws a “cross” to the screen (Image 9), the model seems to think that it is equal to number four after classification. The reason for this being because the model is not trained to detect any other characters than numbers. A possible way to mitigate this issue would be to retrain the model and make it distinguish between single-digit numbers and other characters.

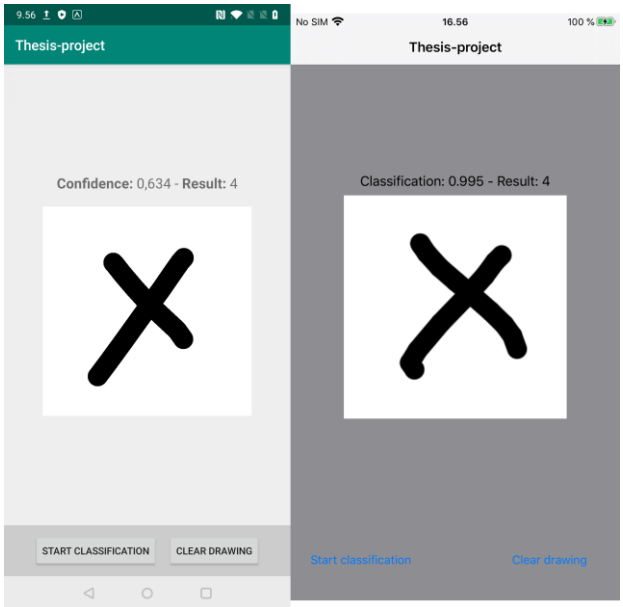

IMAGE 9. Cross (“X”) drawn to the Android and iOS applications

6 CONCLUSION AND DISCUSSION

Because almost every field of study is going towards using machine learning in one way or another, this Bachelors' thesis concludes that knowing about machine learning at a basic level is very useful. The existing resources for gathering fundamental knowledge include various books and internet articles. However, most of those resources seem to be written for data scientists and mathematicians who understand the algorithms behind the machine learning much better than a typical software developer would. In the case that machine learning was explained in more detail, the page count of this thesis would have increased drastically.

This Bachelors' thesis aimed to improve basic knowledge of machine learning-related processes and techniques. All goals set for this thesis were achieved. A theoretical part of this thesis answered the questions which were asked in the beginning. A practical part of the thesis contained a demo project for Android and iOS platforms, which used a machine learning framework to classify handwritten digits. Lastly, because Piceasoft keeps researching new possibilities for machine learning-based products, this thesis should be suitable to act as an instruction manual for those who are unfamiliar with the topic.

Machine learning is a complex yet exciting topic. For the most part, machine learning remains a black box, but developing applications which use machine learning via a framework is not very difficult. The essential software development skills are often good enough. In the grand scheme of things, traditional software development is not going to change much because of machine learning. However, machine learning will make some tasks more straightforward than ever before. Although some cross-platform frameworks can require more tweaking when trying to keep things similar.

The next step would have been to delve deeper into the machine learning territory. For example, improvements to machine learning related terminology would be necessary, and the inclusion of more advanced terminology. Supervised and unsupervised, semi-supervised, and reinforced learning methods must be ex-

plained thoroughly. Also, machine learning algorithms should be discussed extensively. A justification for this extraneous knowledge would be a better understanding of the complete (machine learning) process.

A significant improvement to the demo application would have been to retrain the MNIST model to distinguish between numbers (from zero to nine) but also detect non-numbers. Another possibility would have been to reduce the number of erroneous classifications by disregarding any drawing as non-number if its confidence score is below a threshold value of 55 percent, for example. A final solution to this problem would have been to build a custom-trained model for classifying digits. In this particular case, the final solution comes with a huge downside, which is the time that would need to be used for training the model when considering that the existing MNIST model is already well-trained with tens of thousands of samples, building the own model from the ground up is a waste of time.

As a software developer, a personal opinion of machine learning is that it would require more mathematical background and theoretical thinking to understand well enough. The referenced books and articles were a great resource, but most of them dove too deep into the mathematical/theoretical parts of machine learning. Knowing about the inner-workings of machine learning did not seem beneficial when developing simple applications that use a machine learning framework as a primary goal.

It is reasonable to think that machine learning development keeps getting more accessible in the future because device performance keeps increasing, more machine learning models are shared, and the usability of machine learning frameworks keep improving. It is only a matter of time before we see more sophisticated use cases for machine learning applications.

References

Gopalakrishnan, R. & Venkateswarlu, A. 2018. Machine learning for mobile: practical guide to building intelligent mobile applications powered by machine learning. Birmingham: Packt Publishing.

Image classification | Tensorflow Lite. 2020. Digital article. Modified 4.6.2020. Read 1.8.2020.
https://www.tensorflow.org/lite/models/image_classification/overview

Khandelwal, R. 2020. A Basic Introduction to TensorFlow Lite. Digital article. Published 15.6.2020. Read 1.8.2020.
https://towardsdatascience.com/a-basic-introduction-to-tensorflow-lite-59e480c57292

LeCun Y. & Cortes C. & Burges C. 2013. The MNIST Database of handwritten digits. Published 14.5.2013. Read 27.8.2020.
http://yann.lecun.com/exdb/mnist/

Loukas, S. 2020. What is Machine Learning: Supervised, Unsupervised, Semi-Supervised and Reinforcement learning methods. Digital article. Published 10.6.2020. Read 24.8.2020.
https://towardsdatascience.com/what-is-machine-learning-a-short-note-on-supervised-unsupervised-semi-supervised-and-aed1573ae9bb

Ng, K. 2018. Machine learning projects for mobile applications: build Android and IOS applications using TensorFlow Lite and Core ML. Birmingham: Packt Publishing.

Object detection | Tensorflow Lite. 2020. Digital article. Modified 4.6.2020. Read 1.8.2020.
https://www.tensorflow.org/lite/models/object_detection/overview

Wiggers, K. 2018. Apple's Core ML 2 vs. Google's ML Kit: What's the difference? Digital article. Published 5.6.2018. Read 1.8.2020.
https://venturebeat.com/2018/06/05/apples-core-ml-2-vs-googles-ml-kit-whats-the-difference/

Appendices

Appendix 1. Project source code

The complete source code for Android and iOS mobile applications is publicly available on the GitHub version control service.

Android application source code (written in Java)

https://github.com/Jindetta/Thesis-project-for-Android

iOS application source code (written in Swift)

https://github.com/Jindetta/Thesis-project-for-iOS