# Nowoczesne narzędzia wykorzystywane w technologiach mobilnych

Adam Linke 121529, Mateusz Norel 126901

# Uczenie maszynowe


- iOS
  - CoreML
  - CreateML
- Multiplatformowe
  - Tensorflow Lite
  - MLKit

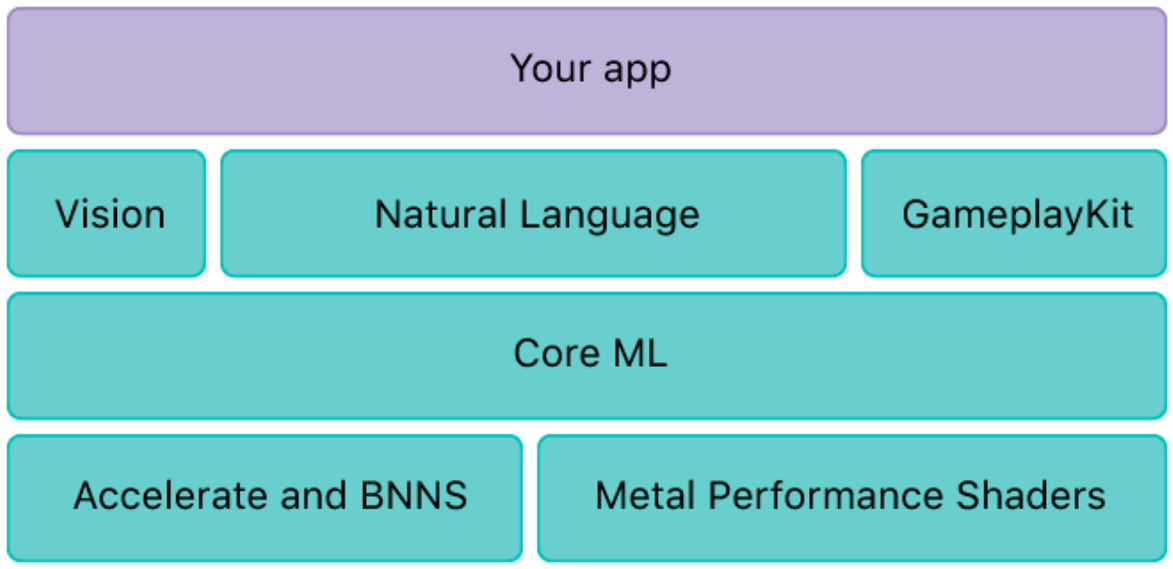
# CoreML

- Narzędzie służące do integracji modelu do aplikacji mobilnej na systemie iOS

- możliwe jest konwertowanie niektórych modeli na zgodne z CoreML takich jak modele:

  - tensorflow

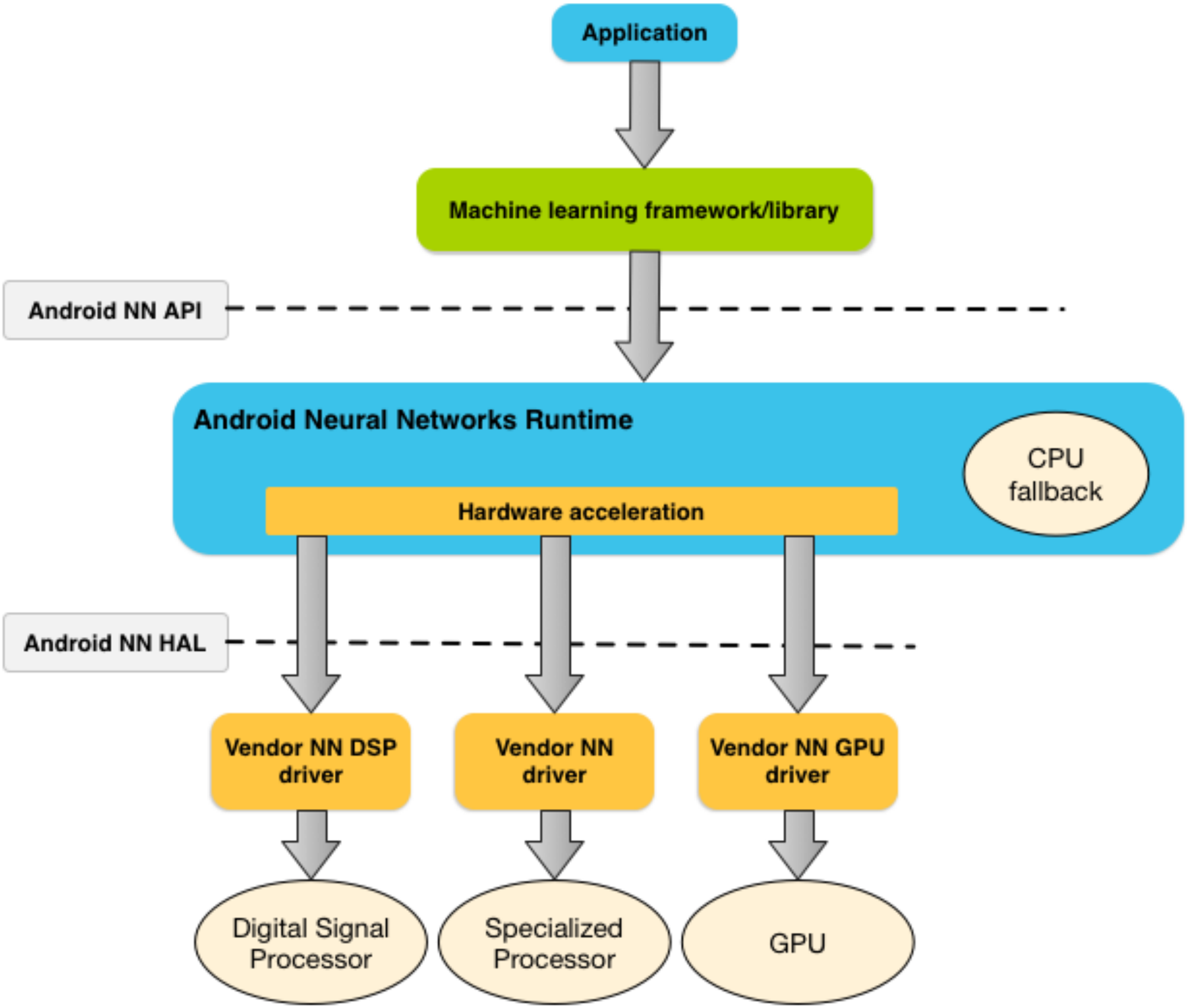
  - MXNet



# CreateML

- Umożliwia bardzo proste tworzenie i uczenie własnych modeli uczenia maszynowego.
- Pozwala na ćwiczenie modeli do zadań takich jak
  - rozpoznawanie obrazów - MLImageClassifierBuilder
  - ekstrahowanie znaczenia z tekstu - MLTextClassifier
  - znajdowania relacji pomiędzy wartościami liczbowymi - MLDataTable
- Oparte o technologie wykorzystywane w Siri i aplikacji Photos(najprawdopodobniej wykorzystuje także idee zawarte w Turi Create)

# Android Neural Networks API



# TensorFlow Lite

- Biblioteka istniejących modeli (Image Classification, Object Detection, Smart Reply, Pose Estimation, Segmentation)
- Narzędzia do konwersji
- Narzędzia do optymalizacji (Pruning, Model Quantization)

# TensorFlow Lite Interpreter

- Android - Java i C++ API
- iOS - Swift i Objective-C API
- Linux - C++ i Python'owe API

# W pełni działające operatory TFLite

- tf.batch_to_space_nd
- tf.exp
- tf.fake_quant
- tf.matmul
- tf.nn.avg_pool
- tf.nn.conv2d
- tf.nn.depthwise_conv2d
- tf.nn.l2_normalize
- tf.nn.local_response_normalization
- tf.nn.log_softmax
- tf.split
- tf.nn.max_pool
- tf.nn.softmax
- tf.nn.top_k
- tf.one_hot
- tf.pad
- tf.reduce_mean
- tf.reshape
- tf.sigmoid
- tf.space_to_batch_nd
- tf.space_to_depth

# Niewspierane (tymczasowo) operatory

- tf.depth_to_space
- tf.image.resize_bilinear
- tf.tanh

Istnieje również część operatorów, która nie jest wspierana, ale jest łatwo zastępowalna, przez np. podstawienie kilku innych operacji. Lista takich operatorów również znajduje się na stronie TensorFlow.

# Konwersja modelu TensorFlow do TensorFlow Lite

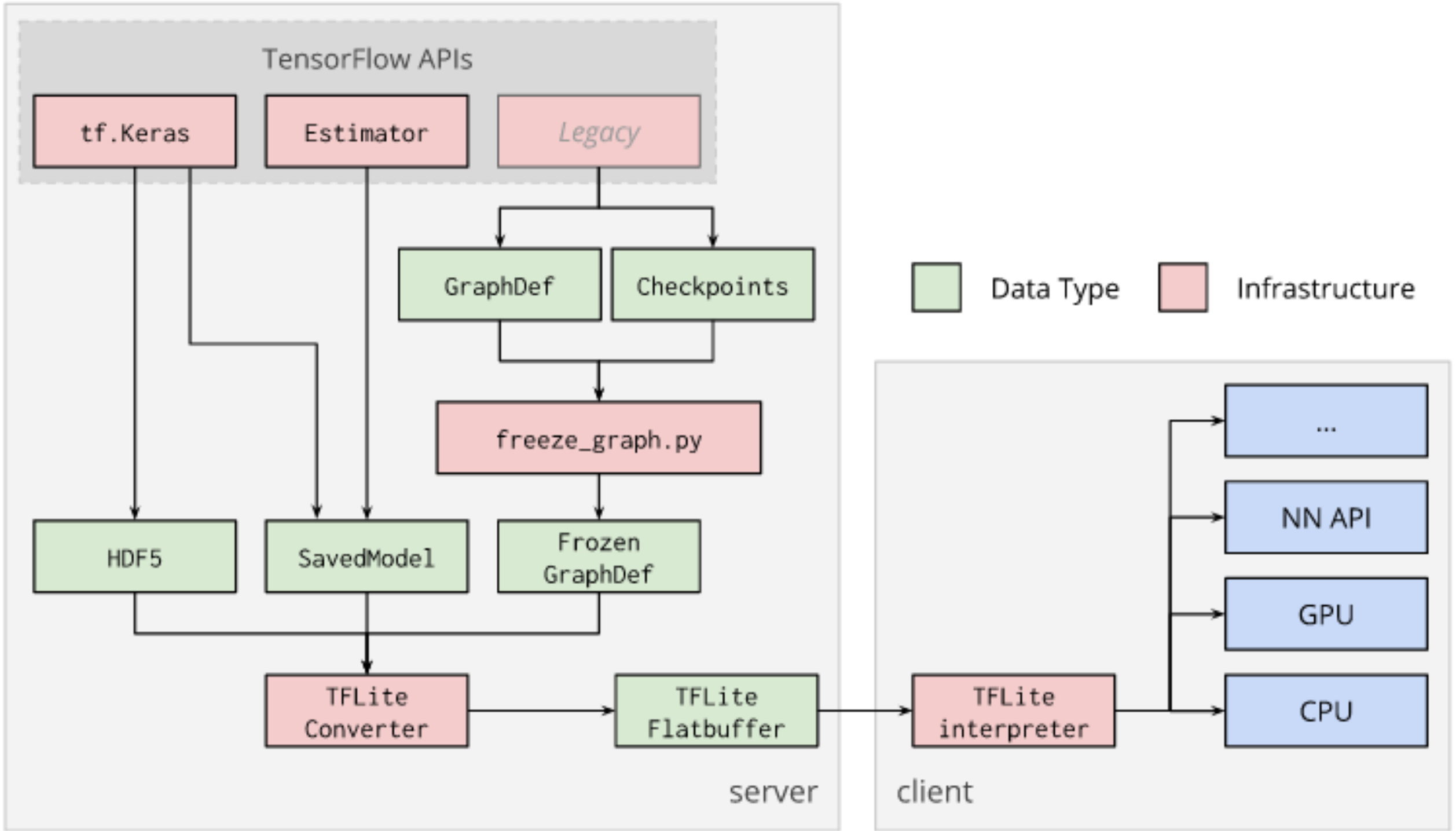
```
tflite_convert \
  --output_file=/tmp/foo.tflite \
  --saved_model_dir=/tmp/saved_model
```

```
tflite_convert \
  --output_file=/tmp/foo.tflite \
  --graph_def_file=/tmp/some_quantized_graph.pb \
  --inference_type=QUANTIZED_UINT8 \
  --input_arrays=input \
  --output_arrays=MobilenetV1/Predictions/Reshape_1 \
  --mean_values=128 \
  --std_dev_values=127
```

```
import tensorflow as tf

converter = tf.lite.TFLiteConverter.from_saved_model(saved_model_dir)
converter.optimizations = [tf.lite.Optimize.OPTIMIZE_FOR_SIZE]
tflite_quant_model = converter.convert()
open("converted_model.tflite", "wb").write(tflite_quantized_model)
```

```
python -m tensorflow.python.tools.optimize_for_inference \
  --input=tf_files/retrained_graph.pb \
  --output=tf_files/optimized_graph.pb \
  --input_names="input" \
  --output_names="final_result"
```

# Schemat konwersji modelu



# Optymalizacja

- Post-training quantization
- Quantization-aware training
- Przycinanie grafu
- zmniejszanie liczby
- optymalizacja dla specjalnych architektur sprzętowych

| Model | Top-1 Accuracy (Original) | Top-1 Accuracy (Post Training Quantized) | Top-1 Accuracy (Quantization Aware Training) | Latency (Original) (ms) | Latency (Post Training Quantized) (ms) | Latency (Quantization Aware Training) (ms) | Size (Original) (MB) | Size (Optimized) (MB) |
|---|---|---|---|---|---|---|---|---|
| Mobilenet-v1-1-224 | 0.709 | 0.657 | 0.70 | 124 | 112 | 64 | 16.9 | 4.3 |
| Mobilenet-v2-1-224 | 0.719 | 0.637 | 0.709 | 89 | 98 | 54 | 14 | 3.6 |
| Inception_v3 | 0.78 | 0.772 | 0.775 | 1130 | 845 | 543 | 95.7 | 23.9 |
| Resnet_v2_101 | 0.770 | 0.768 | N/A | 3973 | 2868 | N/A | 178.3 | 44.9 |

# Użycie modelu

**Java**

```java
try (Interpreter interpreter = new Interpreter(file_of_a_tensorflowlite_model)) {
  interpreter.run(input, output);
}
```

```java
interpreter.runForMultipleInputsOutputs(inputs, map_of_indices_to_outputs);
```

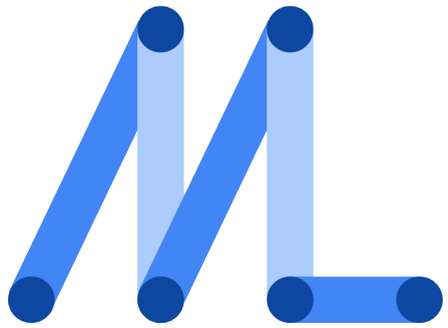
**C++**

```cpp
  tflite::FlatBufferModel model(path_to_model);

tflite::ops::builtin::BuiltinOpResolver resolver;
std::unique_ptr<tflite::Interpreter> interpreter;
tflite::InterpreterBuilder(*model, resolver)(&interpreter);

// Resize input tensors, if desired.
interpreter->AllocateTensors();

float* input = interpreter->typed_input_tensor<float>(0);
// Fill `input`.

interpreter->Invoke();

float* output = interpreter->typed_output_tensor<float>(0);
```

# Przyszłość TFLite

- Więcej operatorów
- Wersjonowanie i sygnatury operatorów
- Nowy konwerter
- Usprawnienie TF Select Ops
- Wsparcie dla LSTM i RNN
- Wizualizacja grafów
- Wsparcie dla technik pre-i-post processingu
- Uczenie na urządzeniu
- Dodatkowe API
- Więcej modeli

# MLKit



- Text Recognition
- Image Labeling
- Face Detection
- Barcode Scanning
- Landmark Recognition

Firebase

# Obliczenia na urządzeniu vs obliczenia w chmurze

| | Zalety | Wady |
|---|---|---|
| **Na urządzeniu** | Szybkość<br>Prywatność<br>Nie wymaga połączenia | Ograniczona złożoność operacji<br>Model zajmuje dodatkowe miejsce |
| **W chmurze** | Złożone operacje<br>Nie potrzeba modelu na urządzeniu<br>Umożliwia zbieranie danych | Większy czas odpowiedzi<br>Wymaga stałego połączenia<br>Potencjalne zagrożenie dla prywatności |

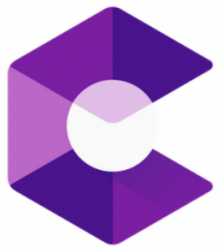
# Czym jest AR

Rozszerzona rzeczywistość(Augmented Reality) - System łączący świat rzeczywisty z generowanym komputerowo. Zazwyczaj wykorzystuje się obraz z kamery, na który nałożona jest, generowana w czasie rzeczywistym, grafika 3D. Istnieją także zastosowania wspomagające jedynie dźwięk.

# Narzędzia do AR

- iOS
  - ARKit
- Android
  - ARCore (ARCore jest też dla iOS)
- Multiplatformowe
  - Vuforia
  - Unity for Mobile AR(wysoko poziomowe(pod spodem ARKit i ARCore))
  - Unreal Engine(wysoko poziomowe(pod spodem ARKit i ARCore))





ARCore

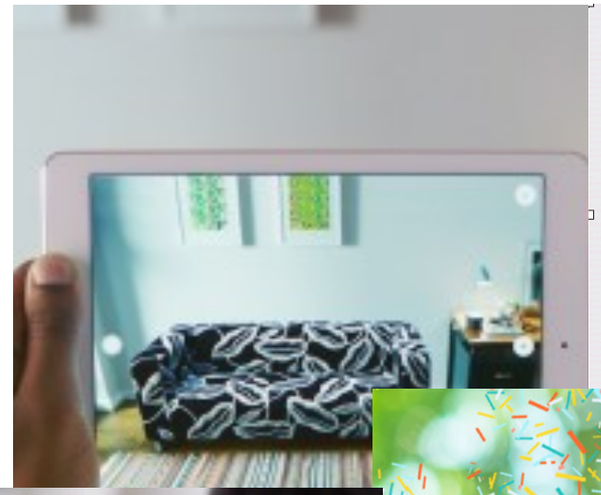
vuforia™

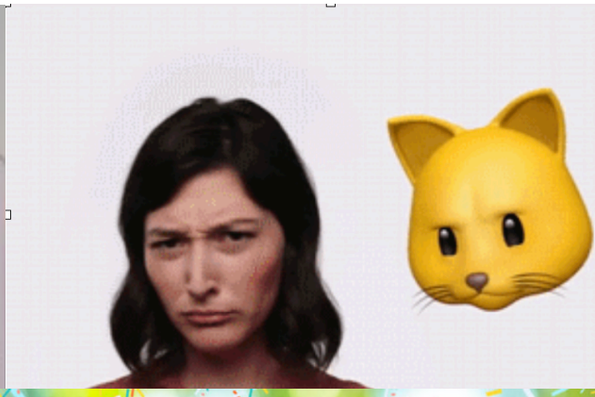
UNREAL
ENGINE

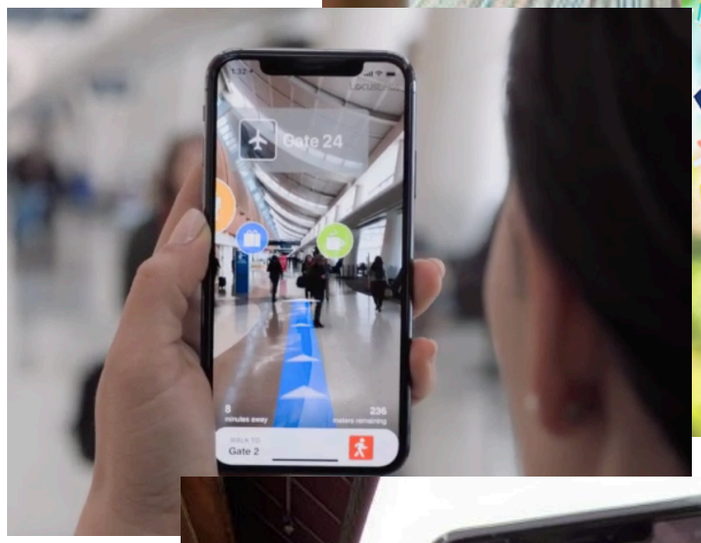
| Narzędzie | Producent | Zastosowanie | Platformy |
|---|---|---|---|
| ARPA SDKs | ARPA Solutions | Komercyjne | Android, iOS (ARPA SDKs), Google Glass (ARPA GLASS SDK), Android, iOS, Windows PC (ARPA Unity Plugin) |
| ARLab SDKs | ARLab | Komercyjne | Android, iOS |
| Metaio SDK | Metaio | Darmowe i komercyjne | Android, iOS, Windows PC, Google Glass, Epson Moverio BT-200, Vuzix M-100, Unity |
| Vuforia SDK | Qualcomm | Darmowe i komercyjne | Android, iOS, Unity |
| Wikitude SDK | Wikitude GmbH | Komercyjne | Android, iOS, Google Glass, Epson Moverio, Vuzix M-100, Optinvent ORA1, PhoneGap, Titanium, Xamarin |

# Dzisiejsze zastosowania AR

- gry (pokemon go, minecraft)
- Architektura i wystrój wnętrz (Ikea Place, Augment)
- Rozpoznawanie twarzy (snapchat)
- Pomiary odległości (Air Measure, MeasureKit, TapMeasure)
- Nawigacja (Google maps, American Airlines)
- Tworzenie wizualizacji sprzętu przemysłowego(GE)







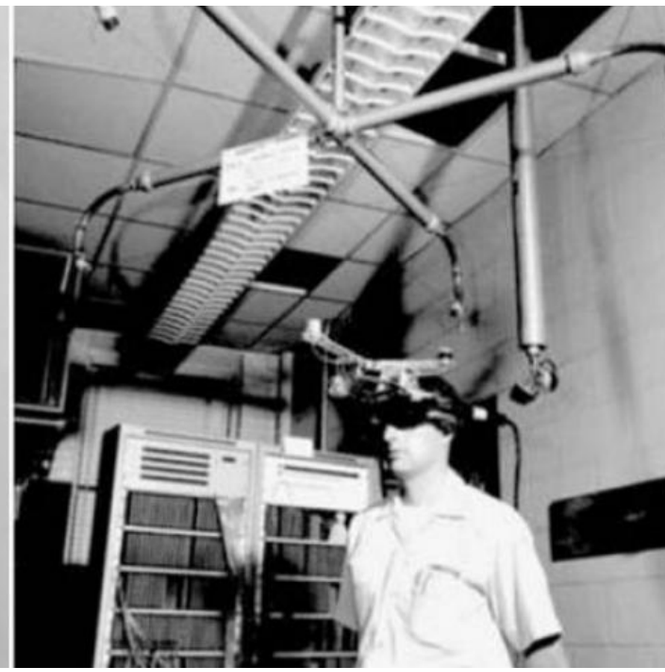

# Ewolucja AR 1

- 1968 - Ivan Sutherland pierwsze urządzenie AR

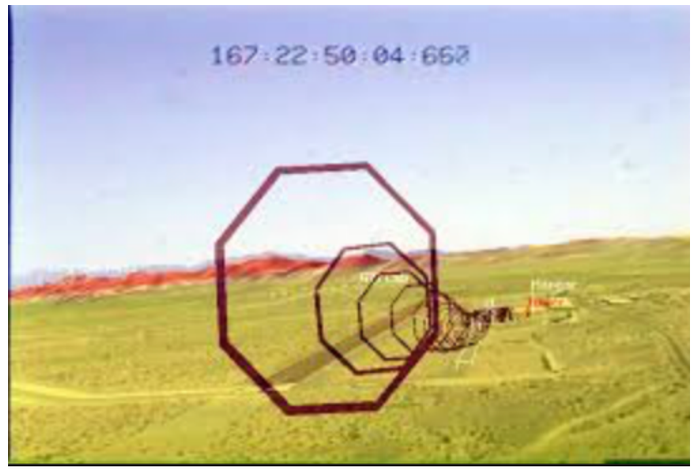
- 1974 - Pierwsze interaktywne środowisko AR

- 1990 - Tom Caudell po raz pierwszy używa terminu "Augmented Reality"

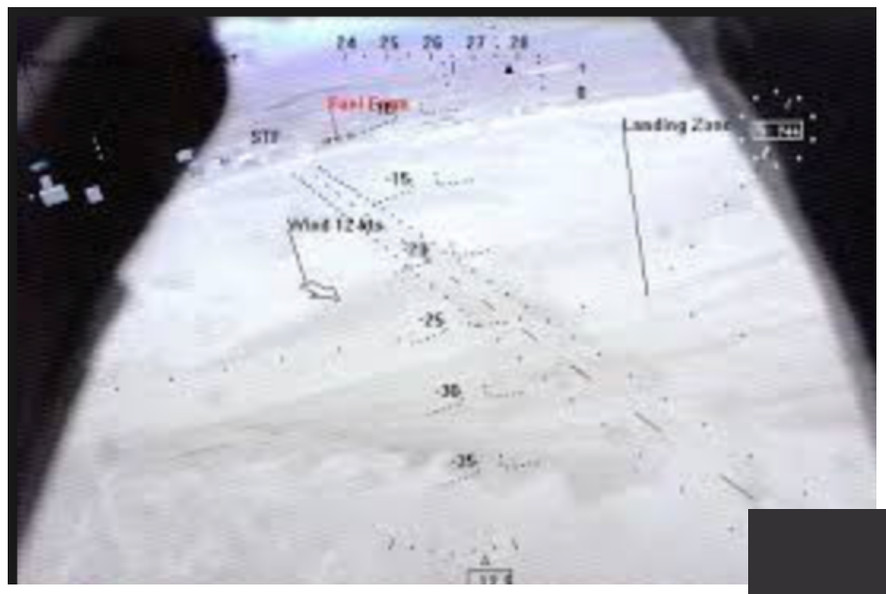
- 1998 - Pojawienie się markerów podczas meczy NFL



# Ewolucja AR 2




- 1999 - NASA X-38
  wykorzystuje system AR aby nakładać dane mapy podczas lotów testowych



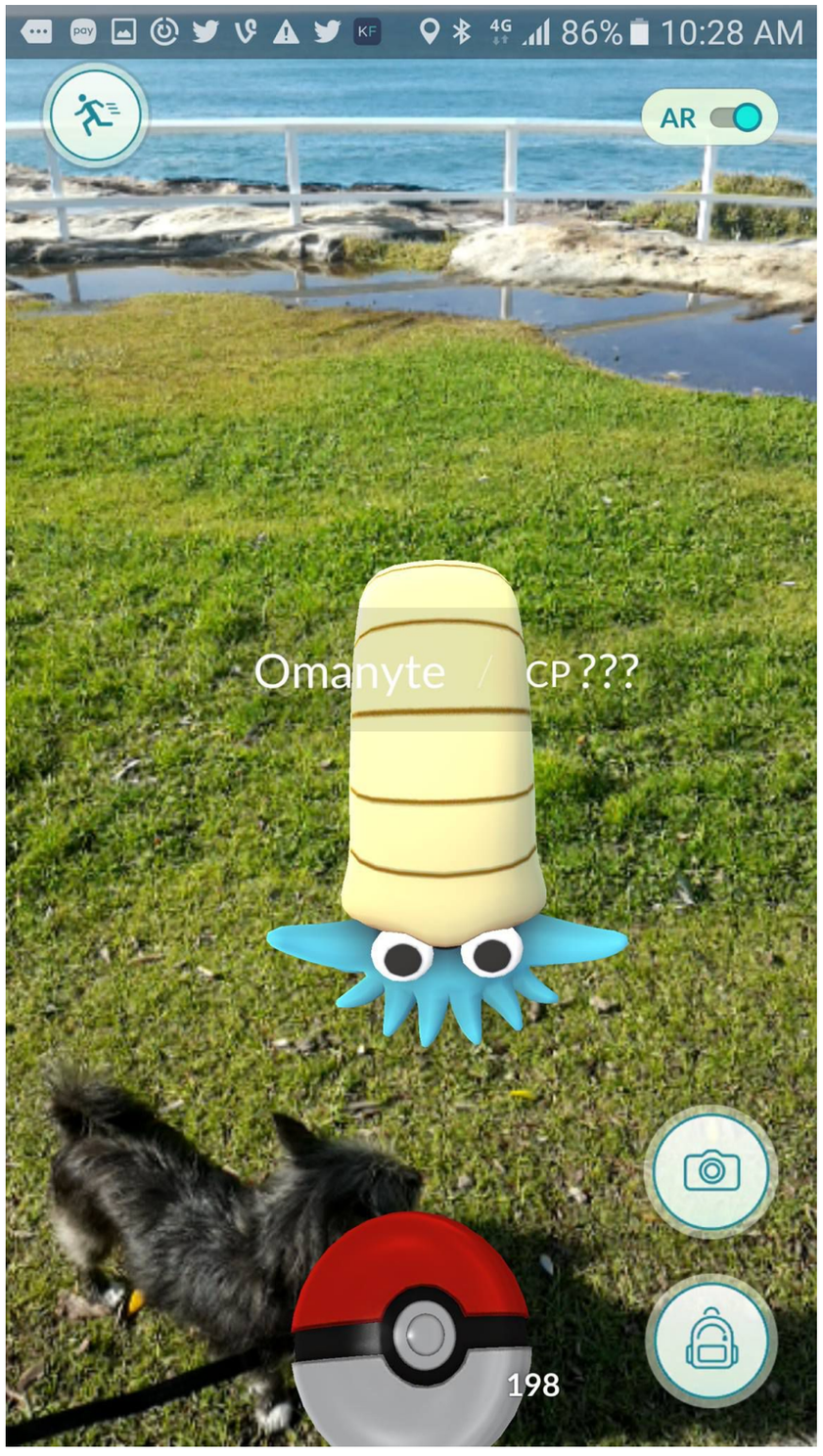
- 2000 - ARToolKit pierwsza biblioteka open-source służąca do nakładania grafik komputerowych na obraz z kamery

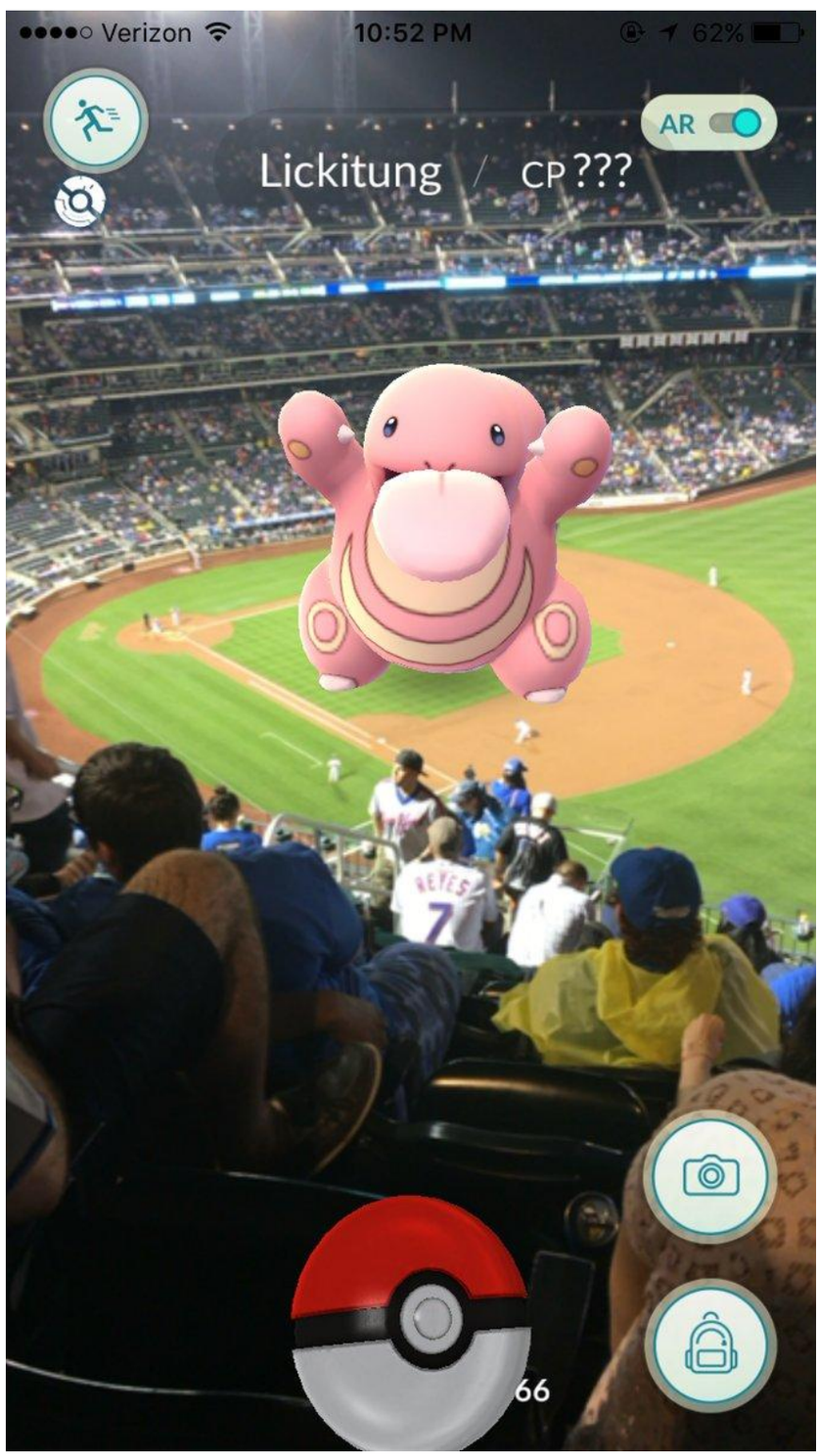
- 2009 - ARToolKit zaczyna być wykorzystywany w przeglądarkach

# Ewolucja AR 3

- 2013 - Producenci samochodów zaczynają wykorzystywać AR do serwisowania samochodów(VW MARTA)
- 2014 - Google Glass
- 2015 - Inwestycje AR wynoszą 700 milionów dolarów
- 2016 - Inwestycje wynoszą 1.1 miliarda dolarów, pojawia się Microsoft HoloLens Developer Kit
- 2017 - premiera ARKit
- 2018 - premiera ARCore i ARKit 2


90% 10:19 AM
AR
Tentacruel / CP ???
199


86% 10:28 AM
AR
Omanyte / CP ???
198


Verizon
10:52 PM
62%
AR
Lickitung / CP ???
66

https://www.youtube.com/watch?v=7ZrmPTPgY3I

# Urządzenia z systemami iOS oraz Android wspierające AR

- iOS
    - iPhone 6s oraz nowsze(w tym także iPhone SE)
    - iPad 5 i 6 generacji oraz iPad Pro(dowolna generacja)
    - nowe modele iPodów
- Android
    - ARCore wymaga systemu Android wersji 7.0 lub późniejszej
    - Wikitude oraz Vuforia wspierają również wcześniejsze wersje Andoid OS (4.4)

# Znajdowanie powierzchni

# Dedykowane urządzenia do AR

- HoloLens
- Atheer AiR Glasses
- DAQRI Smart Glasses
- EPSON Smart Glasses
- Google Glass


# Optymalne warunki do rozszerzonej rzeczywistości

- Powierzchnia posiadająca punkty charakterystyczne(złe: gładkie biurko/podłoga, dobre: podłoga posiadająca różne wzorki/meble)
- dobre oświetlenie, unikanie powierzchni błyszczących
- unikanie szybkich nagłych ruchów aparatem

# Parę przemyśleń po implementacji paru projektów AR na potrzeby prezentacji

- większość testów była wykonywana na starych urządzaniach AR są one bardzo nieprecyzyjne potrafią przesuwać obiekty
- Często aby dane miejsce zostało uznane za powierzchnie musi być ono puste i duże. (W mniejszych pomieszczeniach czasem nie da się znaleźć takiej powierzchni nie powiodły się także testy w których próbowano wykorzystywać drzwi jako taką przestrzeń).

# Przyszłość

- Okulary AR(google glass)
- rosnące inwestycje w technologie AR
- możliwa częściowa dezaktualizacja prezentacji(wwdc w dniu prezentacji o 19:00 najprawdopodobniej rozszerzone zostaną technologie CreateML, CoreML, ARKit)

# Bibliografia


- https://unity3d.com/how-to/create-AR-games-in-Unity-efficiently
- https://developers.google.com/ar/
- https://pl.wikipedia.org/wiki/Rzeczywistość_rozszerzona
- https://www.apple.com/lae/ios/augmented-reality/
- https://developer.apple.com/documentation/
- "App development with Swift" Apple
- https://github.com/tensorflow/examples/tree/master/lite/examples
- https://www.tensorflow.org/lite/