

# Klasyfikacja obrazów w C++

Michał Góraj 171937

Celem projektu jest stworzenie klasyfikatora obrazów, potrafiącego rozróżnić zdjęcia psów od kotów. Rolę tę pełnić będzie konwolucyjna sieć neuronowa, napisana w C++ przy użyciu TensorFlow API.

## Środowisko

Środowisko zostało zainstalowane na Ubuntu 20.04. Sama instalacja TensorFlow i jego konfiguracja tak, by można było korzystać z niego w całości pisząc przy użyciu języka C++ sprawia pewne problemy. Po pierwsze na oficjalnej stronie internetowej tensorflow.org nie ma już bezpośredniej instrukcji o tym mówiącej (kiedyś była, co sygnalizują linki wstawiane przez użytkowników np. na StackOverflow). Użytkownik powinien domyśleć się, że właściwa instrukcja dla niego w tej sytuacji to nie np. wiązania dla języka C, a jedynie *budowanie ze źródła* [1].

```
mordoksieznik@mordoksieznik-MRC-WX0:~/Code$ git clone https://github.com/tensorflow/tensorflow.git
Cloning into 'tensorflow'...
remote: Enumerating objects: 52, done.
remote: Counting objects: 100% (52/52), done.
remote: Compressing objects: 100% (48/48), done.
remote: Total 988171 (delta 16), reused 31 (delta 2), pack-reused 988119
Receiving objects: 100% (988171/988171), 586.75 MiB | 3.03 MiB/s, done.
Resolving deltas: 100% (804819/804819), done.
Updating files: 100% (23280/23280), done.
```

## **Wersja oprogramowania**

Domyślna gałąź rozwojowa projektu to master, dzięki czemu pobieramy najnowszą wersję TensorFlow. Udało mi się przy użyciu obecnej wersji oprogramowania skonfigurować środowisko, napisać pierwszy kod i go odpalić; jednakże w dalszej części pracy w czasie kompilacji pojawiały się niezrozumiałe dla mnie szereg błędów związanych z zawieraniem plików nagłówkowych oprogramowania stron trzecich (eigen, protobuf, brak wymaganej biblioteki absl), po których ręcznej poprawie pojawił się błąd, z którym już nie udało mi się dowiedzieć co należy zrobić. Najlepsze doświadczenie mam z ostatnim oficjalnym wydaniem, tj. wersją z gałęzi r2.3 (git checkout r2.3 w folderze głównym tensorflow).

Z różnymi wersjami TensorFlow również należy uważać – m.in. struktura plików wielokrotnie się zmieniała przez co rozwiązania problemów z Internetu bywają już kompletnie nieprzydatne. Przede wszystkim jednak TensorFlow jest używany w pakiecie z innymi wymaganymi zależnościami, co do których właściwa wersja jest podstawą funkcjonowania. Lista przetestowanych jako kompatybilne wersji kompilatora GCC, Pythona i przede wszystkim Bazela znajduje się na samym dole instrukcji instalacji i należy się jej ściśle trzymać.

## Konfiguracja

Po pobraniu odpowiedniej wersji z Githuba, należy zgodnie z instrukcją uruchomić skrypt `./configure` i za jego pomocą wybrać odpowiednie opcje instalacji. Można tu wybrać możliwość wykorzystania procesora graficznego, czego nie uczyniłem ponieważ uznałem za niepotrzebne do wykonania tego projektu (uczenie nie trwało ostatecznie zbyt długo nawet przy użyciu dość skromnego CPU jakim dysponuję).

```
mordokstieznik@mordokstieznik-MRC-WX0:~/Code/tensorFlow$ ./configure
You have bazel 2.0.0 installed.
Please specify the location of python. [Default is /usr/bin/python]: /usr/bin/python3.8

Found possible Python library paths:
  /usr/local/lib/python3.8/dist-packages
  /usr/lib/python3/dist-packages
Please input the desired Python library path to use. Default is [/usr/local/lib/python3.8/dist-packages]

Do you wish to build TensorFlow with OpenCL SYCL support? [y/N]: N
No OpenCL SYCL support will be enabled for TensorFlow.

Do you wish to build TensorFlow with ROCm support? [y/N]: N
No ROCm support will be enabled for TensorFlow.

Do you wish to build TensorFlow with CUDA support? [y/N]: N
No CUDA support will be enabled for TensorFlow.

Do you wish to download a fresh release of clang? (Experimental) [y/N]: N
Clang will not be downloaded.

Please specify optimization flags to use during compilation when bazel option "--config=opt" is specified [Default is -march=native -Wno-sign-compare]:

Would you like to interactively configure ./.WORKSPACE for Android builds? [y/N]: N
Not configuring the WORKSPACE for Android builds.

Preconfigured Bazel build configs. You can use any of the below by adding "--config=<" to your build command. See .bazelrc for more details.
--config=mkl          # Build with MKL support.
--config=monolithic   # Config for mostly static monolithic build.
--config=ngraph        # Build with Intel nGraph support.
--config=numa          # Build with NUMA support.
--config=dynamic_kernels # (Experimental) Build kernels into separate shared objects.
--config=v2            # Build TensorFlow 2.x instead of 1.x.
Preconfigured Bazel build configs to DISABLE default on features:
--config=noaws        # Disable AWS S3 filesystem support.
--config=nogcp         # Disable GCP support.
--config=nohdfs        # Disable HDFS support.
--config=nonccl        # Disable NVIDIA NCCL support.
Configuration finished
```

## Instalacja Bazela

Wydawać by się mogło, że teraz wystarczy napisać kod wykorzystujący tzw. niskopoziomowe API TensorFlow, zawierając odpowiednie nagłówki dyrektywą `#include` i wszystko powinno działać jak należy. Jednakże owych plików nagłówkowych obecnie nie ma wcale w folderach w których powinny się znajdować.

TensorFlow używa narzędzia Bazel do „automatyzacji tworzenia oprogramowania” i jego użycie jest wymagane do doprowadzenia środowiska do prawidłowego stanu. Instrukcja każe odpalić bazela tak:

```
bazel build //tensorflow/tools/pip_package:build_pip_package
```

z odpowiednimi opcjami. Jest to jak się wydaje stosunkowo dobre rozwiązanie, jeżeli ktoś nie chce później z tego narzędzia korzystać, przekopiuje sobie odpowiednie pliki do właściwych folderów include itd. i będzie pisać kod w zwyczajowy sposób. Jednakże z czasem może okazać się, że jeszcze jakieś szczególne pliki należy wygenerować przy użyciu Bazela itp. Drugim sposobem jest stworzenie sobie folderu dla projektu w strukturze tensorflow i stworzeniu tam odpowiedniego pliku BUILD, służącemu programowi Bazel do kompilacji. Ja tworzyłem swój projekt w folderze tensorflow/tensorflow/examples, gdzie przebywał obok dostarczonych przez producenta przykładowych kodów. W przypadku takiej struktury, zbudowanie środowiska wg reguły z instrukcji nie jest dobrym pomysłem, ponieważ nie będzie ona trzymana w pamięci podręcznej (*ang. cache*) lokalnego serwera Bazela naszego projektu. To sprawi, że powtórzy on całą operację.

Sama instalacja Bazela jako konieczna do „zbudowania” TensorFlow jest wskazana na początku instrukcji z oficjalnej strony. Wskazane jest również użycie Bazeliska, dzięki któremu sama instaluje się właściwa wersja Bazela, kompatybilna z używaną wersją Tensorflow (najnowsza z przedziału podanego w odpowiednim pliku). Można również stworzyć plik .bazelversion w którym zapiszemy sam numer wersji, którą chcemy użyć (gdy środowisko przestało współpracować w ten sposób właśnie zmieniłem wersję używanego Bazela na inną z przedziału uznanych za kompatybilne. Wyeliminowało to jeden rodzaj błędu, ale zrodziło kolejne.)

```
mordoksieznik@mordoksieznik-MRC-WX0:~$ sudo npm install -g @bazel/bazelisk
[sudo] password for mordoksieznik:
/usr/local/bin/bazelisk -> /usr/local/lib/node_modules/@bazel/bazelisk/bazelisk.js
/usr/local/bin/bazel -> /usr/local/lib/node_modules/@bazel/bazelisk/bazelisk.js
+ @bazel/bazelisk@1.6.1
added 1 package in 2.237s
```

## Użycie Bazela – plik BUILD

Najbardziej podstawowa wersja pliku BUILD wygląda tak:

```
load("//tensorflow:tensorflow.bzl", "tf_cc_binary")
tf_cc_binary(
    name = "my_project",
    srcs = ["main.cc"],
    deps = ["//tensorflow/core:tensorflow"]
)
```

Obecnie wszystkie zależności do wczytania należy dodawać się za pomocą zmiennej `tf_cc_binary`. Własne pliki do kompilacji dodaje się po przecinku do atrybutu `srcs` (również pliki nagłówkowe), a wszystkie zależności do `deps`. Przy użyciu Bazela należy oprócz dodania odpowiednich nagłówków, dodać odpowiednie argumenty w pliku BUILD. Przykładowo gdy chcemy użyć funkcji zadeklarowanej w pliku `standard_ops.h`, należy:

1. dodać do kodu dyrektywę:

```
#include "tensorflow/cc/ops/standard_ops.h"
```

2. dodać do pliku BUILD w regule `tf_cc_binary`, w argumencie `deps`:

```
"//tensorflow/cc:cc_ops"
```

By dowiedzieć się jaki argument jest oczekiwany przez Bazela należy sprawdzić który folder ze ścieżki do oczekiwanego pliku nagłówkowego jest najgłębszy spośród tych, w których znajduje się plik BUILD, a następnie znaleźć nazwę reguły, która go wczytuje.

Dla Bazela korzeniem jest folder, w którym znajduje się plik WORKSPACE; w wypadku TensorFlow znajduje się on w „pierwszym” folderze tensorflow. W podanym przypadku sytuacja wygląda tak:

1. W folderze

```
/(ścieżka do TensorFlowa ...)/tensorflow/tensorflow/cc/ops
```

nie ma pliku BUILD

## 2. W folderze

`/(ścieżka do TensorFlowa ...)/tensorflow/tensorflow/cc`

jest plik BUILD, w którym istnieje reguła `cc_ops`, wczytująca plik `standard_ops.h`.

W pliku BUILD w naszym projekcie, do atrybutu `deps` dodajemy ścieżkę do tego folderu i odpowiednią regułę po dwukropku.

## Użycie Bazela – kompilacja

Po utworzeniu folderu z plikami z kodem gotowym do kompilacji i właściwego pliku BUILD, należy wywołać poniższą komendę, będąc w folderze-rodzicu naszego folderu projektu:

```
bazel build nasz_projekt/...
```

Jeżeli wszystko zostało poprawnie skonfigurowane, Bazel skompiluje zależności i nasz program. Zużywa on jednak dużo pamięci RAM; kilkakrotnie zdarzyło mi się, że doprowadził on do zawieszenia komputera i konieczności resetu poprzez zajęcie całej dostępnej pamięci. Instrukcja mówi dodaniu opcji: `--local_ram_resources=2048`, w Internecie można znaleźć inne podobne; nie zauważyłem, aby były one skuteczne. Jedyny działający sposób wg mojego doświadczenia to opcja `--jobs`, ograniczająca liczbę wątków dostępnych dla Bazela. (Dla moich 8Gb RAM żadna liczba większa niż 2 nie jest wskazana).

```
mordoksieznik@mordoksieznik-MRC-WX0:~/code/tensorflow$ bazel build tensorflow/examples/label_image/... --jobs=2
Starting local Bazel server and connecting to it...
... still trying to connect to local Bazel server after 10 seconds ...
INFO: Options provided by the client:
  Inherited 'common' options: --isatty=1 --terminal_columns=101
INFO: Reading rc options for 'build' from /home/mordoksieznik/code/tensorflow/.bazelrc:
  Inherited 'common' options: --experimental_repo_remote_exec
INFO: Reading rc options for 'build' from /home/mordoksieznik/code/tensorflow/.bazelrc:
  'build' options: --apple_platform_type=macos --define framework_shared_object=true --define open_source_build=true --java_toolchain=/third_party/toolchains/java:tf_java_toolchain --host_java_toolchain=/third_party/toolchains/java:tf_java_toolchain --define=tensorflow_enable_mlir_generated_gpu_kernels=0 --define=use_fast_cpp_protos=true --define=allow_oversize_protos=true --spawn_strategy=standalone -c opt --announce_rc --define=grpc_no_ares=true --no incompatible_remove_legacy_whole_archive --no incompatible_prohibit_aapt1 --enable_platform_specific_config --config=short_logs --config=v2
INFO: Reading rc options for 'build' from /home/mordoksieznik/code/tensorflow/.tf_configure.bazelrc:
  'build' options: --action_env PYTHON_BIN_PATH=/usr/bin/python3 --action_env PYTHON_LIB_PATH=/usr/lib/python3/dist-packages --python_path=/usr/bin/python3 --config=xla --action_env TF_CONFIGURE_IOS=0
INFO: Found applicable config definition build:short_logs in file /home/mordoksieznik/code/tensorflow/.bazelrc: --output_filter=DONT_MATCH_ANYTHING
INFO: Found applicable config definition build:v2 in file /home/mordoksieznik/code/tensorflow/.bazelrc: --define=tf_api_versions=2 --action_env=TF2_BEHAVIOR=1
INFO: Found applicable config definition build:xla in file /home/mordoksieznik/code/tensorflow/.bazelrc: --define=with_xla_support=true
INFO: Found applicable config definition build:linux in file /home/mordoksieznik/code/tensorflow/.bazelrc: --copt=-w --host_copt=-w --define=PREFIX=/usr --define=LIBDIR=$(PREFIX)/lib --define=INCLUDEDIR=$(PREFIX)/include --cxxopts=-std=c++14 --host_cxxopts=-std=c++14 --config=dynamic_kernels
INFO: Found applicable config definition build:dynamic_kernels in file /home/mordoksieznik/code/tensorflow/.bazelrc: --define=dynamic_loaded_kernels=true --copt=-DAUTOLOAD_DYNAMIC_KERNELS
INFO: Analyzed 2 targets (365 packages loaded, 31143 targets configured).
INFO: Found 2 targets...
INFO: Deleting stale sandbox base /home/mordoksieznik/.cache/bazel/_bazel_mordoksieznik/076c1de242afa6995a4368baae8c717/sandbox
[4,35] /s/ops 2 actions running
  Compiling tensorflow/core/kernels/training_ops.cc: 32s local
  Compiling tensorflow/core/kernels/record_input_ops.cc: 1s local
```

Wywołanie instrukcji wygląda tak w trakcie budowania:

A tak po udanym zakończeniu:

```
mordoksieznik@mordoksieznik-MRC-WX0:~/code/tensorflow$ bazel build tensorflow/examples/label_image/... --jobs=2
Starting local Bazel server and connecting to it...
... still trying to connect to local Bazel server after 10 seconds ...
INFO: Options provided by the client:
  Inherited 'common' options: --isatty=1 --terminal_columns=101
INFO: Reading rc options for 'build' from /home/mordoksieznik/code/tensorflow/.bazelrc:
  Inherited 'common' options: --experimental_repo_remote_exec
INFO: Reading rc options for 'build' from /home/mordoksieznik/code/tensorflow/.bazelrc:
  'build' options: --apple_platform_type=macos --define framework_shared_object=true --define open_source_build=true --java_toolchain=/third_party/toolchains/java:tf_java_toolchain --host_java_toolchain=/third_party/toolchains/java:tf_java_toolchain --define=tensorflow_enable_mlir_generated_gpu_kernels=0 --define=use_fast_cpp_protos=true --define=allow_oversize_protos=true --spawn_strategy=standalone -c opt --announce_rc --define=grpc_no_ares=true --no incompatible_remove_legacy_whole_archive --no incompatible_prohibit_aapt1 --enable_platform_specific_config --config=short_logs --config=v2
INFO: Reading rc options for 'build' from /home/mordoksieznik/code/tensorflow/.tf_configure.bazelrc:
  'build' options: --action_env PYTHON_BIN_PATH=/usr/bin/python3 --action_env PYTHON_LIB_PATH=/usr/lib/python3/dist-packages --python_path=/usr/bin/python3 --config=xla --action_env TF_CONFIGURE_IOS=0
INFO: Found applicable config definition build:short_logs in file /home/mordoksieznik/code/tensorflow/.bazelrc: --output_filter=DONT_MATCH_ANYTHING
INFO: Found applicable config definition build:xla in file /home/mordoksieznik/code/tensorflow/.bazelrc: --define=tf_api_versions=2 --action_env=TF2_BEHAVIOR=1
INFO: Found applicable config definition build:linux in file /home/mordoksieznik/code/tensorflow/.bazelrc: --define=with_xla_support=true
INFO: Found applicable config definition build:dynamic_kernels in file /home/mordoksieznik/code/tensorflow/.bazelrc: --copt=-w --host_copt=-w --define=PREFIX=/usr --define=LIBDIR=$(PREFIX)/lib --define=INCLUDEDIR=$(PREFIX)/include --cxxopts=-std=c++14 --host_cxxopts=-std=c++14 --config=dynamic_kernels
INFO: Found applicable config definition build:dynamic_kernels in file /home/mordoksieznik/code/tensorflow/.bazelrc: --define=dynamic_loaded_kernels=true --copt=-DAUTOLOAD_DYNAMIC_KERNELS
INFO: Analyzed 2 targets (365 packages loaded, 31143 targets configured).
INFO: Found 2 targets...
INFO: Deleting stale sandbox base /home/mordoksieznik/.cache/bazel/_bazel_mordoksieznik/076c1de242afa6995a4368baae8c717/sandbox
INFO: Elapsed time: 4977.612s, Critical Path: 165.43s
INFO: 1520 processes: 1520 local.
INFO: Build completed successfully, 1525 total actions
mordoksieznik@mordoksieznik-MRC-WX0:~/code/tensorflow$
```

(Przykładowy kod label\_image z repozytorium TensorFlow)

W przypadku użycia innej wersji kompilatora w trakcie tej operacji może pojawić się olbrzymia ilość ostrzeżeń i innych komunikatów.

```
mordoksieznik@mordoksieznik-MRC-WX0: ~/tensorflow

33 | typedef eigen_packet_wrapper<_m28l, 23> Packet16q8l;
    | ^
./third_party/eigen3/unsupported/Eigen/CXX11/src/FixedPoint/PacketMathAVX2.h:34:41: warning: ignoring attributes on template argument '_m28l' (aka '__vector(2) long long int') [-Wignored-attributes]
34 | typedef eigen_packet_wrapper<_m28l, 25> Packet16q8u;
    | ^
./third_party/eigen3/unsupported/Eigen/CXX11/src/FixedPoint/PacketMathAVX2.h:35:41: warning: ignoring attributes on template argument '_m28l' (aka '__vector(2) long long int') [-Wignored-attributes]
35 | typedef eigen_packet_wrapper<_m28l, 26> Packet8q16l;
    | ^
./third_party/eigen3/unsupported/Eigen/CXX11/src/FixedPoint/PacketMathAVX2.h:36:41: warning: ignoring attributes on template argument '_m256l' (aka '__vector(4) long long int') [-Wignored-attributes]
36 | typedef eigen_packet_wrapper<_m256l, 27> Packet8q32l;
    | ^
./third_party/eigen3/unsupported/Eigen/CXX11/src/FixedPoint/PacketMathAVX2.h:37:41: warning: ignoring attributes on template argument '_m28l' (aka '__vector(2) long long int') [-Wignored-attributes]
37 | typedef eigen_packet_wrapper<_m28l, 28> Packet4q32l;
    | ^
INFO: From Compiling tensorflow/core/ops/boosted_trees_ops.cc:
In file included from ./third_party/eigen3/unsupported/Eigen/CXX11/FixedPoint:41,
  from ./tensorflow/core/framework/numeric_types.h:24,
  from ./tensorflow/core/framework/allocator.h:26,
  from ./tensorflow/core/framework/tensor.h:23,
  from ./tensorflow/core/framework/attr_value_util.h:24,
  from ./tensorflow/core/framework/node_def_util.h:22,
  from ./tensorflow/core/framework/shape_inference.h:21,
  from ./tensorflow/core/framework/common_shape_fns.h:20,
  from tensorflow/core/ops/boosted_trees_ops.cc:18:
./third_party/eigen3/unsupported/Eigen/CXX11/src/FixedPoint/PacketMathAVX2.h:30:41: warning: ignoring attributes on template argument '_m256l' (aka '__vector(4) long long int') [-Wignored-attributes]
30 | typedef eigen_packet_wrapper<_m256l, 21> Packet16q16l;
    | ^
./third_party/eigen3/unsupported/Eigen/CXX11/src/FixedPoint/PacketMathAVX2.h:31:41: warning: ignoring attributes on template argument '_m256l' (aka '__vector(4) long long int') [-Wignored-attributes]
31 | typedef eigen_packet_wrapper<_m256l, 21> Packet16q16l;
    | ^
./third_party/eigen3/unsupported/Eigen/CXX11/src/FixedPoint/PacketMathAVX2.h:32:41: warning: ignoring attributes on template argument '_m256l' (aka '__vector(4) long long int') [-Wignored-attributes]
32 | typedef eigen_packet_wrapper<_m256l, 22> Packet32q8u;
    | ^
./third_party/eigen3/unsupported/Eigen/CXX11/src/FixedPoint/PacketMathAVX2.h:33:41: warning: ignoring attributes on template argument '_m28l' (aka '__vector(2) long long int') [-Wignored-attributes]
33 | typedef eigen_packet_wrapper<_m28l, 23> Packet16q8l;
    | ^
./third_party/eigen3/unsupported/Eigen/CXX11/src/FixedPoint/PacketMathAVX2.h:34:41: warning: ignoring attributes on template argument '_m28l' (aka '__vector(2) long long int') [-Wignored-attributes]
34 | typedef eigen_packet_wrapper<_m28l, 25> Packet16q8u;
    | ^
./third_party/eigen3/unsupported/Eigen/CXX11/src/FixedPoint/PacketMathAVX2.h:35:41: warning: ignoring attributes on template argument '_m28l' (aka '__vector(2) long long int') [-Wignored-attributes]
35 | typedef eigen_packet_wrapper<_m28l, 26> Packet8q16l;
    | ^
./third_party/eigen3/unsupported/Eigen/CXX11/src/FixedPoint/PacketMathAVX2.h:36:41: warning: ignoring attributes on template argument '_m256l' (aka '__vector(4) long long int') [-Wignored-attributes]
36 | typedef eigen_packet_wrapper<_m256l, 27> Packet8q32l;
    | ^
./third_party/eigen3/unsupported/Eigen/CXX11/src/FixedPoint/PacketMathAVX2.h:37:41: warning: ignoring attributes on template argument '_m28l' (aka '__vector(2) long long int') [-Wignored-attributes]
37 | typedef eigen_packet_wrapper<_m28l, 28> Packet4q32l;
    | ^
Target //tensorflow/tools/pip_package:build_pip_package up-to-date:
  bazel-bin/tensorflow/tools/pip_package/build_pip_package
INFO: Elapsed time: 13133.052s, Critical Path: 1714.12s
INFO: 9188 processes: 9188 local.
INFO: Build completed successfully, 9952 total actions
mordoksieznik@mordoksieznik-MRC-WX0:~/tensorflow$
```



Widać również, że `build_pip_package` trwa znacznie dłużej – w moim przypadku ponad 3,5 godziny zamiast około 80 minut.

## Dane

Dane wejściowe stanowi mała wersja bazy z Kaggle [2]. Zawiera ona zdjęcia kotów i psów, uporządkowanych w folderach `train`, `validate` i `test`. W każdym z tych trzech folderów znajdują się podfoldery `cats` i `dogs`, a w nich po 1000 zdjęć kotów i 1000 zdjęć psów do uczenia sieci oraz po 500 dla walidacji oraz testowania. Wszystkie zdjęcia są zapisane w formacie `jpg`.



Przykładowe zdjęcie kota z bazy danych

# Kod

## TensorFlow C++ API

Programowanie z użyciem API TensorFlow odbywa się w dwóch etapach. Najpierw tworzy się tzw. *graf* (*ang. graph*) dla którego operacje są wierzchołkami a dane wejściowe i wyjściowe krawędziami. Graf jest modelem, zapisującym zależności między przewidzianymi operacjami. Czynność, która wygląda jak normalne wywołanie funkcji jest dopiero dodaniem operacji do grafu. Drugi etap to samo wywołanie owego grafu w ramach tzw. *sesji* (*ang. session*), czyli połączenia z silnikiem TensorFlow. W ramach wywołania przekazuje się mu dane wejściowe, zbiór elementów do ewaluacji i kontener na wyniki. Grafy i *podgrafy* (*ang. subgraphs*) znajdują się w tzw. *zasięgu* (*ang. scope*) – obiekcie, który przechowuje cały tzw. kontekst operacji, same grafy i niektóre fizyczne zasoby.

Kluczowym pojęciem jest tytułowy tensor – uogólniony wektor (np. wektor jest tensorem 1. rzędu, a macierz 2.). W nich na ogół przechowywane są dane wejściowe i wyjściowe z kolejnych operacji. Tensor używany w programie do reprezentacji serii (*ang. batch*) obrazów jest rzędu czwartego i zapisany w tzw. formacie NHWC. Wymiary reprezentują tu po kolei: numer obrazu w serii, wysokość, szerokość i liczbę kanałów (3, format RGB).

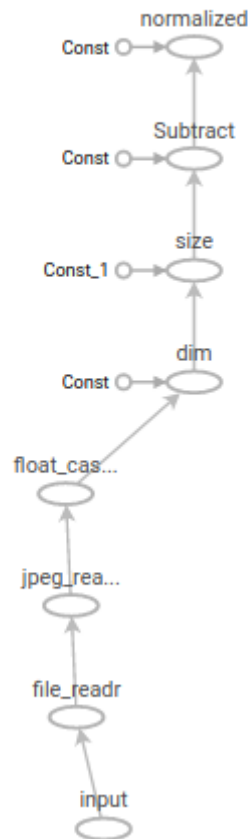
Kod użyty w ramach projektu stworzony został na podstawie serii poradników, znajdujących się pod przypisem [3].

## **Wczytywanie danych**

Sam kod zawiera dużą liczbę komentarzy, wyjaśniających sens działania danej funkcji czy danego fragmentu kodu. Funkcja `CreateGraphForImage` tworzy graf operacji służących do wczytania obrazu z pliku do odpowiedniego tensora trzeciego rzędu. Funkcja `ReadTensorFromImageFile` służy do ewaluacji takiego grafu. `ReadFileTensors` używa powyższej funkcji do wczytania wszystkich obrazów z danego folderu. Ostatecznie funkcja `ReadBatches` wykorzystuje `ReadFileTensors` i następnie przetwarza jej rezultaty, by połączyć serie w pojedyncze tensory czwartego rzędu w formacie NHWC.

Działanie całego programu jest koordynowane w funkcji `main`. Wywołanie `CreateGraphForImage` odbywa się tylko raz, dopóki sam graf nie musi zostać zmieniony. Następnie dla trybów pracy wykorzystujących serie obrazów (uczenie i walidacja) używana jest funkcja `ReadBatches`; dla testowania, gdzie obrazy wczytywane są pojedynczo, potrzebna jest modyfikacja grafu i ponowne wywołanie `CreateGraphForImage`, a następnie `ReadFileTensors`.

## Main Graph



Graf stworzony przez funkcję CreateGraphForImage

### Sieć neuronowa

Kluczową funkcją dla tworzenia modelu konwolucyjnej sieci neuronowej jest CreateGraphForCNN. To tu definiowana jest jej architektura.

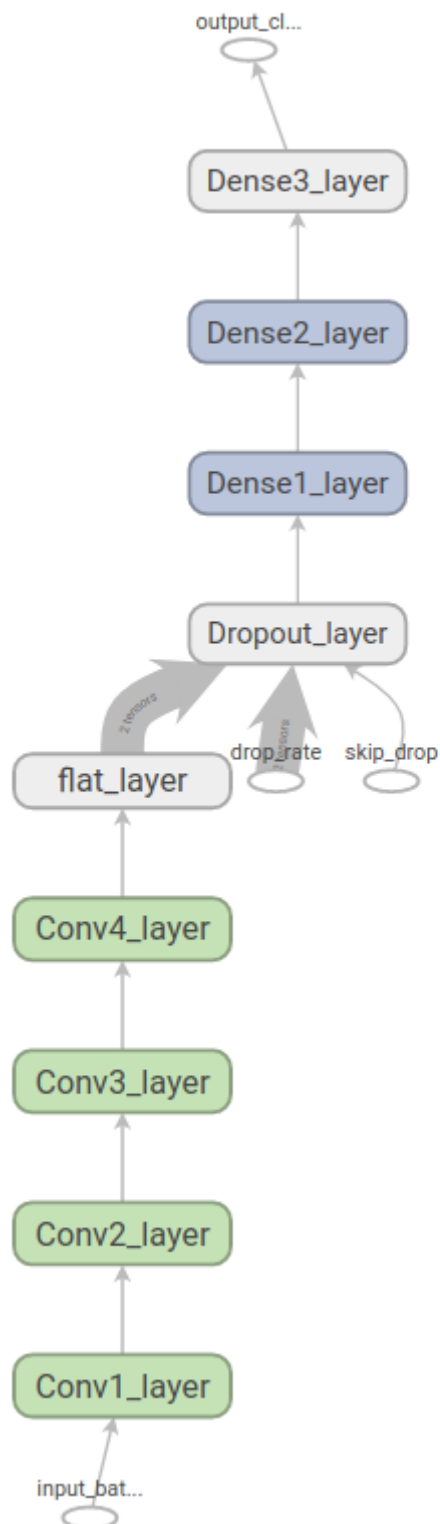
Użyty model zawiera 4 warstwy konwolucyjne, z filtracją o kroku (*ang. stride*) równym 1 i otoczką (*ang. padding*) 1 (określaną jako *ang. same-size*, zachowującą rozmiar). W kodzie każda warstwa konwolucyjna połączona jest z tzw. warstwą łączącą (*ang. pooling layer*), wydzielaną przez teorię. Łączenie odbywa się poprzez wyznaczanie maksimum (*ang. maxpooling*) o kroku 2 i oknie 2x2.

Dalej w sieci odbywa się spłaszczenie danych (*ang. flatten*), by dostosować je do znajdujących się trzech warstw gęstych (*ang. dense layer*). Jako funkcji aktywacyjnej używa się w nich tzw. ReLU (*ang. Rectified linear units*) – poza ostatnią, gdzie do podania ostatecznego wyniku używa się funkcji sigmoidalnej. Używa się również techniki losowego wyłączania neuronów (*ang. dropout*) z warstwy gęstej.

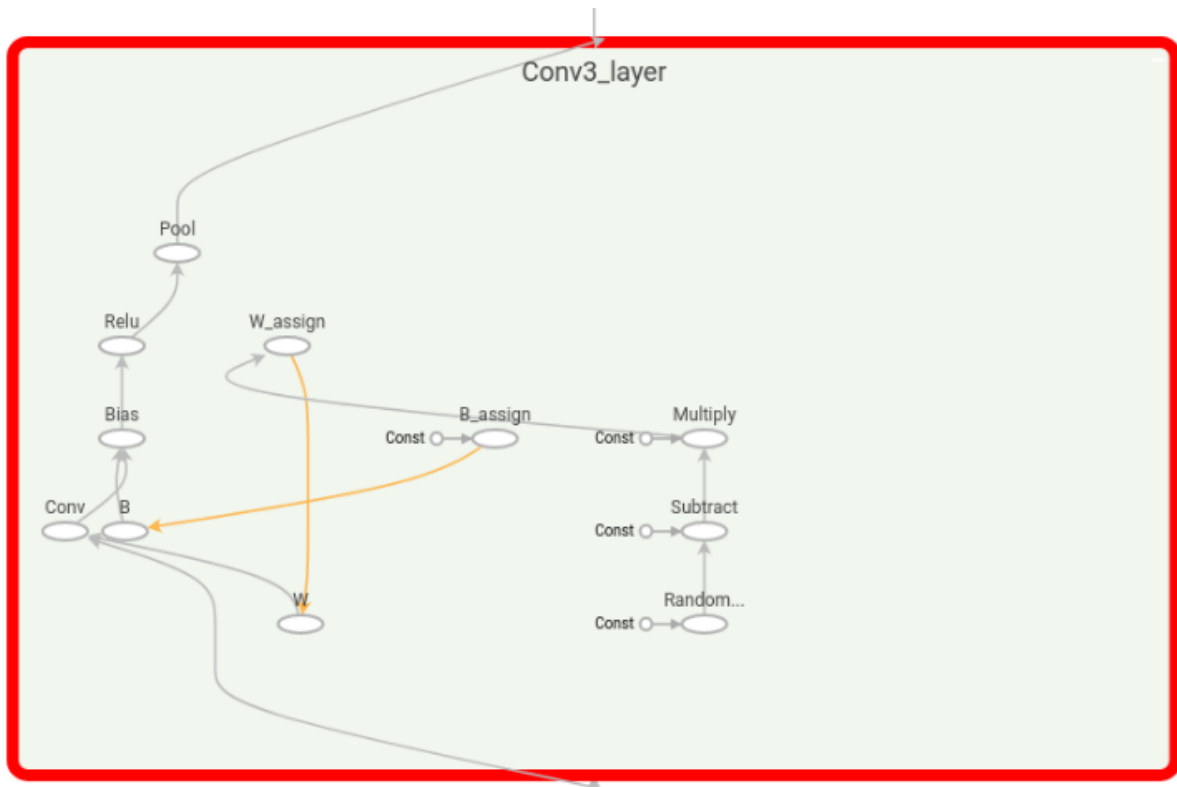
Funkcje `AddConvolutionLayer` i `AddDenseLayer` służą do stworzenia odpowiednio warstwy konwolucyjnej i gęstej. Dodają również do grafu operacje inicjalizacji, które dla wag odbywają się zgodnie z metodą Xaviera. Do map zbiera się zmienne do owej inicjalizacji oraz wagi (*ang. weight*) i skosy (*ang. bias*) wraz z ich wymiarami (*ang. shape*) do propagacji wstecznej (*ang. backpropagation*).

W main cały ten etap jest realizowany przez wywołanie funkcji `CreateGraphForCNN`.

## Main Graph



Graf konwolucyjnej sieci neuronowej (bez propagacji wstecznej)

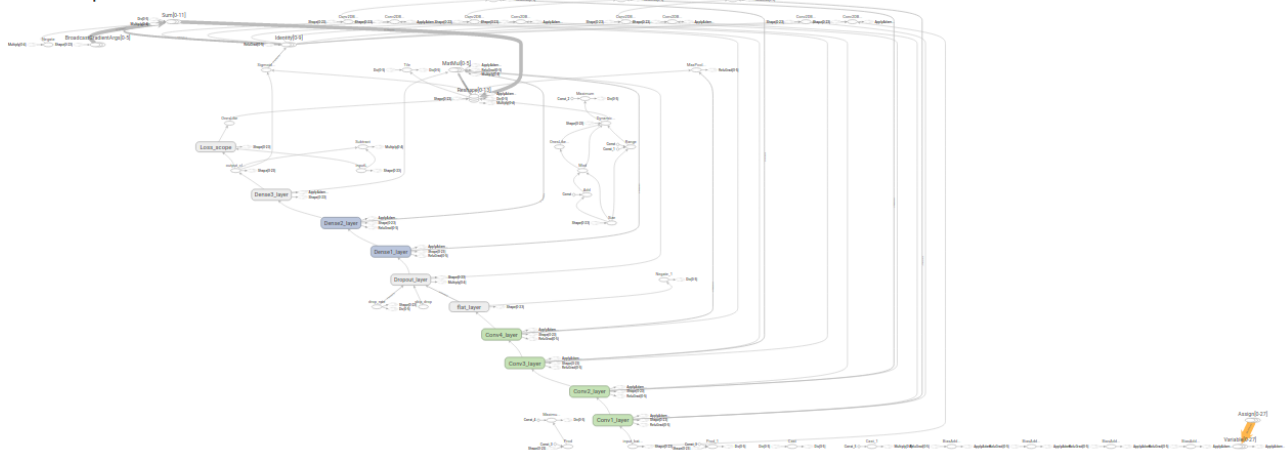


Podgraf warstwy konwolucyjnej nr 3

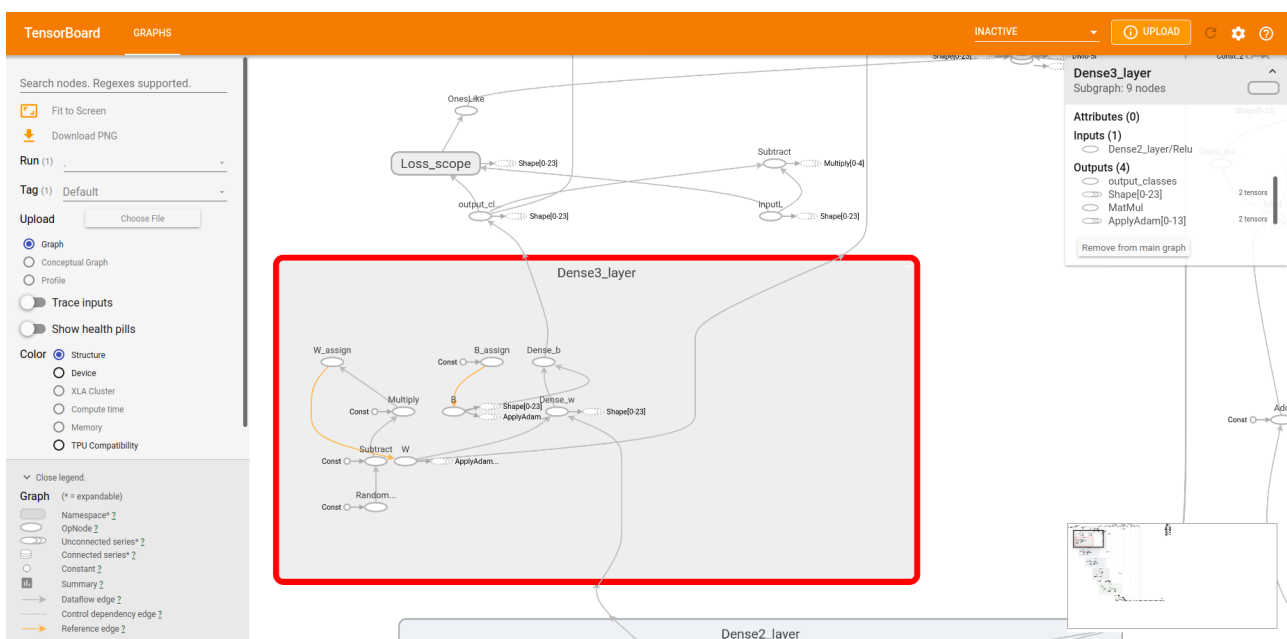
## Propagacja wsteczna

CreateOptimizationGraph rozszerza graf stworzonej sieci o elementy propagacji wstecznej. W tym celu wobec rezultatów klasyfikacji podjętej przez sieć obliczania jest funkcja kosztu – średnia kwadratów różnic między odpowiedzią sieci a oczekiwanym rezultatem. Następnie do grafu dodawana jest operacja licząca gradienty, wskazujące kierunek najszybszego wzrostu/spadku wartości funkcji kosztu. Dalej jako metodę użycia informacji z obliczonych gradientów do wyznaczenia zmienionych wartości wag i skosów użyty jest algorytm optymalizacyjny Adam. Funkcja CreateOptimizationGraph również wywoływana jest w main.

Main Graph



Graf konwolucyjnej sieci neuronowej z propagacją wsteczną



Podgraf przykładowej warstwy CNN

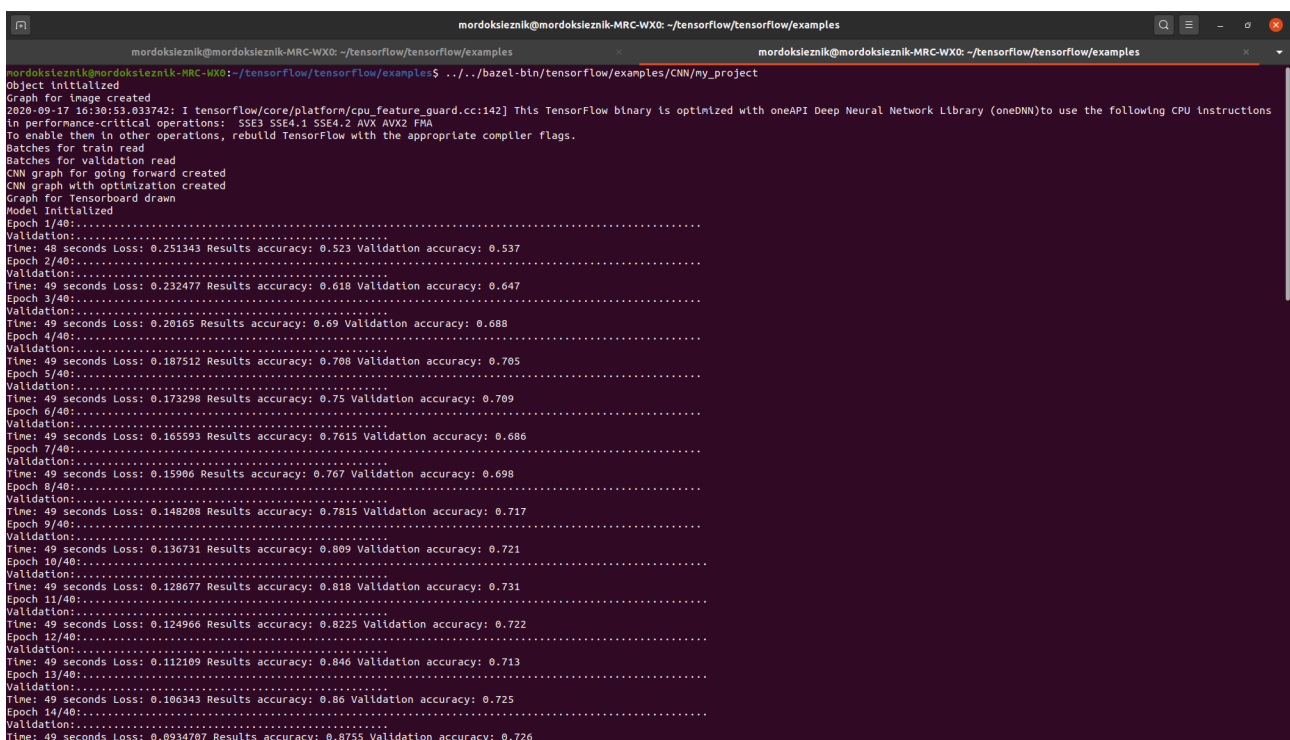


## Uczenie sieci

Po wczytaniu danych wejściowych i stworzeniu grafu sieci z propagacją wsteczną pozostało jedynie uruchomienie sesji z odpowiednią konfiguracją i sprawdzenie rezultatów. Funkcje `TrainCNN`, `ValidateCNN` i `Predict` służą właśnie do uruchomienia sesji w której wykonywana będzie klasyfikacja przez sieć neuronową w odpowiednich trybach. W czasie uczenia obowiązywało będzie wyłączenie neuronów, a także wyślemy silnikowi wynik funkcji kosztów oraz nowe wartości wag i skosów do obliczenia. Tych dwóch czynności unikniemy w trybach walidacji i predykcji (silnik TensorFlow jest „leniwy” i wykonuje tylko te operacje, które są mu potrzebne do obliczenia wartości elementów, o które został poproszony; dzięki temu nie wykorzysta on danych z walidacji do poprawy wartości swoich wag, bo nie „będzie mu się chciało” liczyć funkcji kosztów, gradientów ani algorytmu optymalizacji Adama), przy czym *dropout* nie jest operacją pominiętą, a taką w której można podać wartości parametrów tak, by ta operacja nie miała żadnego efektu na funkcjonowanie sieci.

Podobnie zdefiniowana jest również funkcja `Initialize` – wywołuje ona sesję w której silnik dokonuje ewaluacji zmiennych inicjowanych zgodnie z podgrafem sieci neuronowej, zapisanych uprzednio do mapy zmiennych do inicjalizacji.

W main dokonujemy inicjalizacji m.in. wag sieci CNN, a następnie uruchamiamy uczenie dla kolejnych serii danych. Pojedyncza klasyfikacja każdego obrazu z puli stanowi jedną epokę (*ang. epoch*) nauki. Po dokonaniu takiego treningu, sieć dokonuje walidacji – próbnej klasyfikacji zbioru do tego służącego (w ramach tej samej epoki). Cała operacja jest następnie powtarzana przez zadaną liczbę epok (20). Na końcu dokonywana jest predykcja, gdzie wczytywane są pojedynczo obrazy testowe, sprawdzające efektywność sieci. Zarówno wyniki poszczególnych epok uczenia, jak i testów wyświetlane są w oknie konsoli.



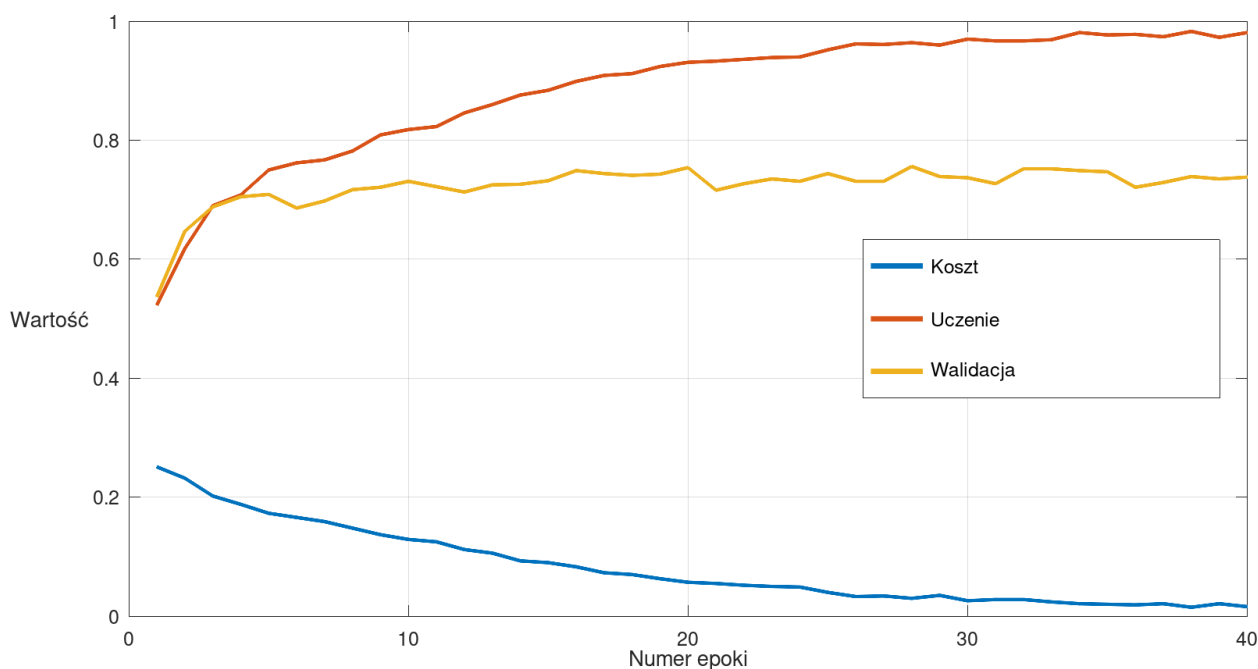
```
mordoksieznik@mordoksieznik-MRC-WX0: ~/tensorflow/tensorflow/examples
mordoksieznik@mordoksieznik-MRC-WX0: ~/tensorflow/tensorflow/examples
mordoksieznik@mordoksieznik-MRC-WX0: ~/tensorflow/tensorflow/examples
mordoksieznik@mordoksieznik-MRC-WX0: ~/tensorflow/tensorflow/examples$ ../../bazel-bin/tensorflow/examples/CNN/my_project
Object initialized
Graph for image created
2020-09-17 16:30:53.033743: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions
in performance-critical operations: SSE3 SSE4.1 SSE4.2 AVX AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Batches for train read
Batches for validation read
CNN graph for going forward created
CNN graph with optimization created
Graph for Tensorboard drawn
Model initialized
Epoch 1/40:.....
Validation:.....
Time: 48 seconds Loss: 0.251343 Results accuracy: 0.523 Validation accuracy: 0.537
Epoch 2/40:.....
Validation:.....
Time: 49 seconds Loss: 0.232477 Results accuracy: 0.618 Validation accuracy: 0.647
Epoch 3/40:.....
Validation:.....
Time: 49 seconds Loss: 0.20165 Results accuracy: 0.69 Validation accuracy: 0.688
Epoch 4/40:.....
Validation:.....
Time: 49 seconds Loss: 0.173298 Results accuracy: 0.75 Validation accuracy: 0.709
Epoch 5/40:.....
Validation:.....
Time: 49 seconds Loss: 0.187512 Results accuracy: 0.708 Validation accuracy: 0.705
Epoch 6/40:.....
Validation:.....
Time: 49 seconds Loss: 0.165593 Results accuracy: 0.7615 Validation accuracy: 0.686
Epoch 7/40:.....
Validation:.....
Time: 49 seconds Loss: 0.15986 Results accuracy: 0.767 Validation accuracy: 0.698
Epoch 8/40:.....
Validation:.....
Time: 49 seconds Loss: 0.148208 Results accuracy: 0.7815 Validation accuracy: 0.717
Epoch 9/40:.....
Validation:.....
Time: 49 seconds Loss: 0.136731 Results accuracy: 0.809 Validation accuracy: 0.721
Epoch 10/40:.....
Validation:.....
Time: 49 seconds Loss: 0.128677 Results accuracy: 0.818 Validation accuracy: 0.731
Epoch 11/40:.....
Validation:.....
Time: 49 seconds Loss: 0.124966 Results accuracy: 0.8225 Validation accuracy: 0.722
Epoch 12/40:.....
Validation:.....
Time: 49 seconds Loss: 0.112109 Results accuracy: 0.846 Validation accuracy: 0.713
Epoch 13/40:.....
Validation:.....
Time: 49 seconds Loss: 0.106343 Results accuracy: 0.86 Validation accuracy: 0.725
Epoch 14/40:.....
Validation:.....
Time: 49 seconds Loss: 0.0934707 Results accuracy: 0.8755 Validation accuracy: 0.726
```

```
mordoksieznik@mordoksieznik-MRC-WX0: ~/tensorflow/tensorflow/examples
mordoksieznik@mordoksieznik-MRC-WX0: ~/tensorflow/tensorflow/examples

Epoch 15/40:.....
Validation:.....
Time: 49 seconds Loss: 0.0900153 Results accuracy: 0.884 Validation accuracy: 0.732
Epoch 16/40:.....
Validation:.....
Time: 49 seconds Loss: 0.0827409 Results accuracy: 0.899 Validation accuracy: 0.749
Epoch 17/40:.....
Validation:.....
Time: 49 seconds Loss: 0.0728764 Results accuracy: 0.909 Validation accuracy: 0.744
Epoch 18/40:.....
Validation:.....
Time: 49 seconds Loss: 0.0702906 Results accuracy: 0.9115 Validation accuracy: 0.741
Epoch 19/40:.....
Validation:.....
Time: 49 seconds Loss: 0.0628646 Results accuracy: 0.9235 Validation accuracy: 0.743
Epoch 20/40:.....
Validation:.....
Time: 49 seconds Loss: 0.0572321 Results accuracy: 0.931 Validation accuracy: 0.754
Epoch 21/40:.....
Validation:.....
Time: 49 seconds Loss: 0.0546013 Results accuracy: 0.9325 Validation accuracy: 0.716
Epoch 22/40:.....
Validation:.....
Time: 49 seconds Loss: 0.0519361 Results accuracy: 0.9355 Validation accuracy: 0.727
Epoch 23/40:.....
Validation:.....
Time: 49 seconds Loss: 0.0495938 Results accuracy: 0.939 Validation accuracy: 0.735
Epoch 24/40:.....
Validation:.....
Time: 49 seconds Loss: 0.0486507 Results accuracy: 0.94 Validation accuracy: 0.731
Epoch 25/40:.....
Validation:.....
Time: 49 seconds Loss: 0.039081 Results accuracy: 0.9515 Validation accuracy: 0.744
Epoch 26/40:.....
Validation:.....
Time: 49 seconds Loss: 0.0330629 Results accuracy: 0.962 Validation accuracy: 0.731
Epoch 27/40:.....
Validation:.....
Time: 49 seconds Loss: 0.0337541 Results accuracy: 0.9605 Validation accuracy: 0.731
Epoch 28/40:.....
Validation:.....
Time: 49 seconds Loss: 0.0299151 Results accuracy: 0.9635 Validation accuracy: 0.756
Epoch 29/40:.....
Validation:.....
Time: 49 seconds Loss: 0.0353465 Results accuracy: 0.96 Validation accuracy: 0.739
Epoch 30/40:.....
Validation:.....
Time: 49 seconds Loss: 0.0262876 Results accuracy: 0.97 Validation accuracy: 0.737
Epoch 31/40:.....
Validation:.....
Time: 49 seconds Loss: 0.0276803 Results accuracy: 0.967 Validation accuracy: 0.727
Epoch 32/40:.....
Validation:.....
Time: 49 seconds Loss: 0.02811 Results accuracy: 0.967 Validation accuracy: 0.752
```

```
mordoksieznik@mordoksieznik-MRC-WX0: ~/tensorflow/tensorflow/examples
mordoksieznik@mordoksieznik-MRC-WX0: ~/tensorflow/tensorflow/examples

Validation:.....
Time: 49 seconds Loss: 0.0262876 Results accuracy: 0.97 Validation accuracy: 0.737
Epoch 31/40:.....
Validation:.....
Time: 49 seconds Loss: 0.0276803 Results accuracy: 0.967 Validation accuracy: 0.727
Epoch 32/40:.....
Validation:.....
Time: 49 seconds Loss: 0.02811 Results accuracy: 0.967 Validation accuracy: 0.752
Epoch 33/40:.....
Validation:.....
Time: 49 seconds Loss: 0.0244483 Results accuracy: 0.969 Validation accuracy: 0.752
Epoch 34/40:.....
Validation:.....
Time: 49 seconds Loss: 0.0207776 Results accuracy: 0.9805 Validation accuracy: 0.749
Epoch 35/40:.....
Validation:.....
Time: 49 seconds Loss: 0.0201491 Results accuracy: 0.9765 Validation accuracy: 0.747
Epoch 36/40:.....
Validation:.....
Time: 49 seconds Loss: 0.0185688 Results accuracy: 0.978 Validation accuracy: 0.721
Epoch 37/40:.....
Validation:.....
Time: 49 seconds Loss: 0.0212341 Results accuracy: 0.9735 Validation accuracy: 0.729
Epoch 38/40:.....
Validation:.....
Time: 49 seconds Loss: 0.015324 Results accuracy: 0.983 Validation accuracy: 0.739
Epoch 39/40:.....
Validation:.....
Time: 49 seconds Loss: 0.0208796 Results accuracy: 0.9725 Validation accuracy: 0.735
Epoch 40/40:.....
Validation:.....
Time: 49 seconds Loss: 0.0162905 Results accuracy: 0.981 Validation accuracy: 0.738
Test number: 1 predicted: 1 actual is: 1
Test number: 2 predicted: 1 actual is: 0
Test number: 3 predicted: 1 actual is: 0
Test number: 4 predicted: 1 actual is: 1
Test number: 5 predicted: 1 actual is: 1
Test number: 6 predicted: 0 actual is: 0
Test number: 7 predicted: 1 actual is: 0
Test number: 8 predicted: 0 actual is: 0
Test number: 9 predicted: 1 actual is: 1
Test number: 10 predicted: 0 actual is: 0
Test number: 11 predicted: 1 actual is: 0
Test number: 12 predicted: 1 actual is: 0
Test number: 13 predicted: 1 actual is: 1
Test number: 14 predicted: 1 actual is: 1
Test number: 15 predicted: 0 actual is: 0
Test number: 16 predicted: 1 actual is: 1
Test number: 17 predicted: 1 actual is: 1
Test number: 18 predicted: 0 actual is: 0
Test number: 19 predicted: 1 actual is: 1
Test number: 20 predicted: 0 actual is: 1
total successes: 14 out of 20
mordoksieznik@mordoksieznik-MRC-WX0: ~/tensorflow/tensorflow/examples$
```



Powyższe wyniki naniesione na wykres

Widać, że w wyniku uczenia sieci funkcjonuje prawidłowo – funkcja kosztu maleje, a dokładność ocen dla zbioru uczącego rośnie. W pierwszej fazie uczenia wyniki dla zbioru uczącego i walidacyjnego są bardzo podobne. Jednakże po kilku epokach wyniki dla zbioru walidacyjnego zaczyna odstawać od zbioru treningowego i wartość . Zachodzi przetrenowanie sieci (*ang. overfitting*). Powtórzone wywołania programu daje podobne rezultaty, a najwyższa dokładność jaką udaje się osiągnąć naszej CNN to ok. 75-76%. Podobne wyniki można obserwować w fazie testowania – wyniki tej fazy w kolejnych wywołaniach są z przedziału 14-16 poprawnych wyników na 20.

Jako że dalsze uczenie na obecnym zbiorze nie może już dać poprawy rezultatów, aby poprawić wyniki sieci konwolucyjnej można jedynie zwiększyć zbiór uczący lub zmienić jej strukturę. To zadanie wykracza poza zakres wykonanego projektu.

## **Dodatkowe uwagi**

1. Żeby operacja `AddSymbolicGradients` w czasie uruchomienia skompilowanego programu zamiast wykonania swojej czynności nie wypisała komunikatu podobnego do poniższego:

```
No gradient defined for op: Mean. Please see  
https://www.tensorflow.org/code/tensorflow/cc/gradients/README.md  
for instructions on how to add C++ gradients.
```

należało znaleźć w jednym z plików o rozszerzeniu `cc` z podanego adresu funkcję realizującą operację liczenia gradientu z naszej operacji (w podanym przypadku funkcja `Mean`, której gradient jest liczony w `nn_ops.cc`). Następnie odnaleźć plik nagłówkowy odpowiadający znalezionemu plikowi z kodem i dodać dyrektywę `#include` dla niego. W naszym przypadku

```
#include "tensorflow/cc/ops/nn_ops.h"
```

oraz do atrybutu `deps` w `tf_cc_binary` w pliku `BUILD` naszego projektu dodać `"/tensorflow/cc:gradients"`. Jeżeli dla danej operacji nie ma w tych plikach liczących gradienty, oznacza to że najpewniej nie została ona nigdy napisana. W takiej sytuacji należy albo napisać kod sieci bez użycia takiej metody bądź napisać własną. Pozostałe problemy z konfiguracją Bazela itp. zostały ostatecznie rozwiązane sposobami opisanymi w sekcji konfiguracji.

2. Funkcja `writeGraphForTensorboard` służy do tworzenia i zapisywania do plików reprezentacji graficznej stworzonych w kodzie grafów operacji. Ogląda się je za pośrednictwem serwera TensorBoard, uruchamianego komendą.

```
mordoksieznik@mordoksieznik-MRC-WX0:~/Code/tensorflow/tensorflow/examples$ tensorboard --logdir .  
Serving TensorBoard on localhost; to expose to the network, use a proxy or pass --bind_all  
TensorBoard 2.3.0 at http://localhost:6006/ (Press CTRL+C to quit)
```

Wypisuje ona adres serwera udostępnianego pod `localhostem`, z którym należy połączyć się za pomocą przeglądarki. Stąd pochodzą wizualizacje grafów z tego sprawozdania.

## Źródła

- [1] <https://www.tensorflow.org/install/source>
- [2] [https://s3.amazonaws.com/img-datasets/cats\\_and\\_dogs\\_small.zip](https://s3.amazonaws.com/img-datasets/cats_and_dogs_small.zip)
- [3] <https://itnext.io/creating-a-tensorflow-dnn-in-c-part-1-54ce69bbd586>  
<https://towardsdatascience.com/creating-a-tensorflow-cnn-in-c-part-2-eea0de9dcada>