Nasz program podzieliliśmy na cztery części:

* generate\_data (C++)
* train (C++)
* test (C++)
* report\_generator (Python)

Funkcję łączącą te programy pełnią skrypty konsoli:

* generate\_data.bat
* train.bat
* test.bat
* all.bat

## generate\_data.exe

Program ten zajmuje się generowaniem danych służących do uczenia sieci. Otwiera podany jako parametr plik zawierający zdjęcie tablicy rejestracyjnej i za pomocą przedstawionego wcześniej algorytmu odczytuje z niej znaki. Ważnym założeniem dotyczącym pracy programu jest to, że nazwa pliku wejściowego (z obciętym rozszerzeniem .jpg) musi być taka sama jak numer tablicy rejestracyjnej (np. sg6190h.jpg przedstawia zdjęcie tablicy z numerem SG 6190H).

Jeśli liczba znalezionych przez algorytm znaków zgadza się z liczbą znaków na tablicy, przystępujemy do ich dyskretyzacji. Każdy z obrazków z literą jest skalowany do rozmiaru 8x12 pikseli i zmieniany w zera lub jedynki, na podstawie koloru na obrazku (czarny albo biały). Przykładowy obrazek z literą „S” będzie więc reprezentowany tak (nowe linie i formatowanie dodane dla czytelności):

0 0 0 **1 1** 1 0 0   
0 **1 1 1 1 1 1 1**   
0 **1** 0 0 0 0 0 **1**   
0 **1** 0 0 0 0 0 0   
0 **1** 0 0 0 0 0 0   
0 **1** 0 0 0 0 0 0   
0 0 **1 1 1 1 1** 0   
0 0 0 0 0 0 **1 1**   
0 0 0 0 0 0 0 **1**   
0 0 0 0 0 0 0 **1**   
0 **1** 0 0 0 0 0 **1**   
0 **1 1 1 1 1 1** 0

Jednocześnie program znajduje aktualnie odczytany znak (jaka to znak został odczytana wiemy, ponieważ litery w nazwie pliku przetwarzane są po kolei a znalezione znaki od lewej do prawej) w tablicy wszystkich możliwych do odczytania znaków (od A do Z i od 0 do 9). Następnie wypisuje listę z ustawioną jedynką na pozycji odpowiadającej znakowi. Litera S jest 19 w alfabecie, stąd jedynka jest na 19. pozycji:  
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

W przypadku, gdy algorytm znajdzie inną na zdjęciu inną liczbę znaków niż tyle ile mógłby się spodziewać wnioskując z nazwy pliku, program kończy się błędem i nie są generowane żadne dane wyjściowe.

## generate\_data.bat

W pliku traindata.txt mamy zdefiniowane numery rejestracji, których chcemy użyć do uczenia sieci, zdjęcia z tymi numerami muszą znajdować się w podkatalogu photos/. Skrypt generate\_data.bat wywołuje dla każdej linii z pliku traindata.txt program generate\_data.exe. Wyniki zapisywane są do pliku regplate.train, razem nagłówkiem w którym zawarta jest liczba wygenerowanych danych testowych. Plik ten zgodny jest ze strukturą danych testowych używanych przez FANN, to znaczy:

<liczba danych testowych> <liczba neuronów wejściowych> <liczba neuronów wyj.>  
<pierwszy wiersz z danymi testowymi>  
<pierwszy wiersz z wynikiem>  
<drugi wiersz z danymi testowymi>  
<drugi wiersz z wynikiem>  
itd.

W naszym przypadku wierszami testowymi będą wygenerowane zdyskretyzowane postaci znaków, a wynikami tablica z „1” na pozycji znaku. Liczba neuronów wejściowych jest równa 96 (8x12, czyli tyle ile pikseli na obrazku), a wyjściowych 36 (liczba możliwych znaków).

## train.exe

Program ten otwiera plik regplate.train i tworzy nową sieć, wykorzystując do jej uczenia dane z tego pliku. Biblioteka FANN odczytuje z pliku kolejne wiersze z danymi i oczekiwane rezultaty, generując plik regplate.net, zawierający dane sieci do wykorzystania w przetwarzaniu.

Do uczenia sieci należy podać trzy parametry, decydujące o jej skuteczności: oczekiwany błąd przy którym następuje zakończenie uczenia, liczbę warstw oraz liczbę neuronów ukrytych w tych warstwach. Przykładowe wywołanie train.exe:

train.exe 0.001 4 50 30

Spowoduje uruchomienie algorytmu uczącego sieć o czterech warstwach (z czego druga warstwa ma 50 neuronów, a trzecia 30), aż osiągnie oczekiwany błąd 0.001 (program zostanie przerwany, jeśli ten błąd nie zostanie osiągnięty w trakcie 3000 epok). Podawanie liczby neuronów na pierwszej i ostatniej warstwie nie ma sensu, gdyż zawsze będzie to liczba odczytywana z pliku traindata.txt.

Skrypt train.bat po prostu wywołuje train.exe z zadanymi parametrami.

## test.exe

test.exe podobnie jak generate\_data.exe otwiera podany jako argument wejściowy plik z ze zdjęciem przedstawiającym tablicę rejestracyjną. Następnie uruchamia algorytm znajdujący litery. W przypadku, gdy liczba liter na tablicy nie zgadza się z tą zawartą w nazwie pliku, program kończy się komunikatem o błędzie. W innym wypadku odczytuje sieć z pliku regplate.net i dla każdego fragmentu zdjęcia zawierającego znak uruchamia funkcję fann\_run, próbującą rozpoznać ten znak. Wyniki rozpoznawania zwracane przez FANN wypisywane są w formacie:

Best match for letter S is S with propability 0.832

Biblioteka przypisuje każdej możliwej literze pewne prawdopodobieństwo, że to ona znajduje się na zdjęciu, jednak my wyświetlamy tylko tę o najwyższych szansach. W przypadku, gdy biblioteka zupełnie nie rozpoznała obrazu i prawdopodobieństwo wynosi 0 dla każdego znaku, wyświetlany jest komunikat:

No match for letter S

## test.bat

Skrytp ten wykonuje program test.exe dla wszystkich tablic z pliku testdata.txt i zapisuje jego wyjście do pliku report.txt. Plik ten zawiera wszystkie wyniki testów w prostym formacie tekstowym.

## report\_generator.py

Aby uniknąć żmudnej analizy wszystkich wyników, napisaliśmy skrypt w Pythonie, parsujący plik report.txt i generujący z niego stronę HTML, zawierającą zebrane w przyjazny dla użytkownika programu statystyki na temat badanej sieci w odniesieniu do danych testowych. Strona zawiera informacje o podanych parametrach sieci, statystykę rozpoznanych liter (liczba liter występujących w danych testowych i jaki procent z nich udało się rozpoznać) oraz poszczególne wyniki dla każdej z zadanych tablic. Raport zapisywany jest w pliku report.html.

## all.bat

Jest to skrypt służący do prostego odpalenia po kolei plików generate\_data.bat, train.bat, test.bat i report\_generator.py, pozwalający za pomocą jednego polecenia zbadać wyniki dla kolejnej konfiguracji sieci. Parametry wejściowe:

all.bat <oczekiwany błąd> <liczba warstw> <liczba neuronów na warstwie 2> <l. neur. na warstwie 3> ... <l. neur. na warstwie N - 1>

W wyniku otrzymujemy aktualny plik report.html.