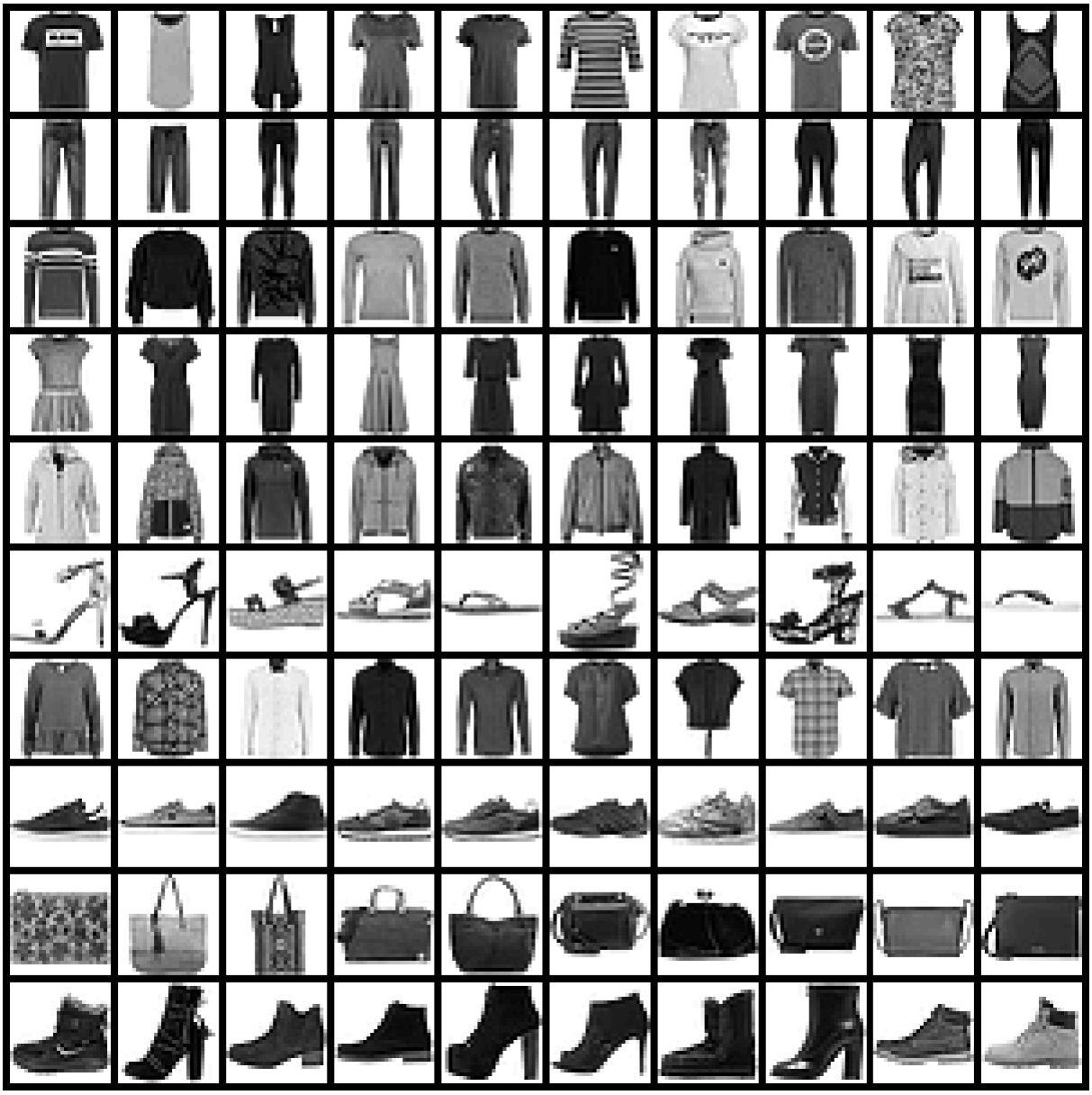
Laboratorium Rozpoznawania Obrazów – Ćwiczenie #5 & #6
Rozpoznawanie odzieży z wykorzystaniem sieci neuronowych

Termin oddawania: **13.05**, **20.05**

W kolejnym ćwiczeniu użyjemy zbioru fashion-MNIST – obrazów 10 rodzajów odzieży zestawionych przez firmę zalando w formacie dokładnie takim, jak odręcznie pisane cyfry, które rozpoznawaliście Państwo w poprzednim ćwiczeniu. Więcej informacji o tym zbiorze znajdziecie Państwo w repozytorium GitHub (https://github.com/zalandoresearch/fashion-mnist). Same dane są też dostępne na serwerze Galera (http://galeranew.ii.pw.edu.pl/~rkz/rob/fashion-mnist.zip).

Dane, z którymi przyjdzie Państwu powalczyć, to: t-shirt/top, spodnie, sweter, sukienka, kurtka/płaszcz, sandały, koszula, trampki, torba, buty.



Rozwiązaniem referencyjnym, z którym można porównywać osiągi swojego rozwiązania jest sieć neuronowa z jedną warstwą ukrytą (100 neuronów) uczona iteracyjnie z ustalonym współczynnikiem uczenia 0.001 przez 50 epok.

	Zbiór uczący fashion-MNIST			Zbiór testowy fashion-MNIST		
	OK.	Błąd	Odrzucenie	OK.	Błąd	Odrzucenie
Jakość klasyfikacji	90.35%	9.65%	0.00%	87.54%	12.46%	0.00%

Wyniki tego pierwszego podejścia nie są rewelacyjne, zatem miejsca na poprawę jest sporo. W rozwiązaniu referencyjnym przyjąłem najprostszy sposób ustalania wyniku klasyfikacji max po wyjściach sieci – stąd brak odrzuceń. Można zastanowić się nad wprowadzeniem dodatkowych warunków przy podejmowaniu decyzji (z myślą o przerzuceniu części błędów do koszyka decyzji wymijających), ale nie jest to element niezbędny. Ważne, żeby podejmowanie decyzji było takie samo w rozwiązaniu referencyjnym i udoskonalonym.

Przyjmijmy, że pozostaniemy przy klasyfikacji przez w pełni połączone wielowarstwowe sieci neuronowe uczonymi algorytmem ze wsteczną propagacją błędu. Dwa podstawowe elementy tego ćwiczenia to:

1. Przygotowanie referencyjnej implementacji uczenia sieci ze wsteczną propagacją błędów (do uzyskania maksymalnej oceny jest konieczna jakość **lepsza**, niż klasyfikatora referencyjnego powyżej).

   Realizacja tego punktu będzie wymagać zapewne od Państwa nieco eksperymentowania z architekturą sieci (liczba warstw, liczba neuronów w warstwach) i parametrami uczenia (wartość i sposób modyfikacji stałej uczenia).

2. Przeczytanie załączonego artykułu Yanna LeCuna i spółki „*Efficient BackProp*" i wybranie z niego modyfikacji sieci lub algorytmu uczenia (np. metody modyfikacji stałej uczenia, kolejności prezentacji próbek…) i przeprowadzenie podobnej, jak w przypadku sieci referencyjnej, serii eksperymentów w poszukiwaniu najlepszego rozwiązania.

W przypadku tego zadania można przyjąć, że kryterium zatrzymania uczenia będzie jakość klasyfikacji zbioru testowego (tzn. użyjemy zbioru testowego w funkcji zbioru walidacyjnego; to oznacza, że będziemy mieć nieco lepsze niż faktycznie wyniki klasyfikacji).

W sprawozdaniu proszę zamieścić:

1. Opis metody uczenia ze szczególnym uwzględnieniem wariantu niestandardowego.
2. Dane dotyczące liczby epok i czasów uczenia.
3. Macierze pomyłek i ich analizę dla rekomendowanej sieci.
4. Oczywiście oprócz sprawozdania przesyłacie Państwo kod oraz **parametry generatora liczb losowych** (patrz niżej) – chodzi o to, żebym miał możliwość wygenerowania dokładnie takich samych sieci, jak opisane w sprawozdaniu.

   Do pobrania stanu generatora losowego należy użyć polecenia:

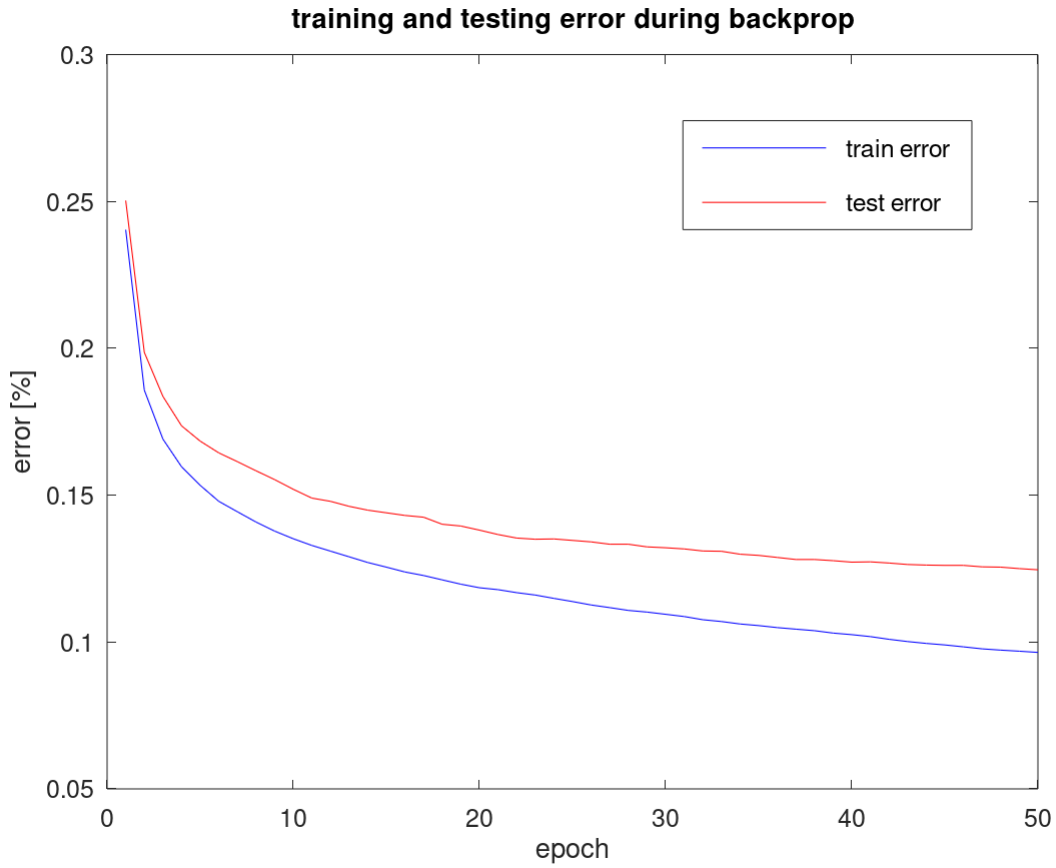
   ```
   rand();
   rstate = rand(”state”);
   ```

   i tak otrzymany wektor zapisać:

   ```
   save rnd_state.txt rstate
   ```

Do pakietu sprawozdanie + kod proszę nie dołączać danych (koszt dołączenia danych wynosi 2 punkty)!

Poniższy wykres przedstawia współczynniki błędów mierzone w trakcie uczenia sieci.



Nie widać na nim żadnych dramatycznych zmian, ale kończymy z siecią przeuczoną ☹

Legenda kodu:

actdf.m	-	pochodna funkcji aktywacji
actf.m	-	funkcja aktywacji
anncls.m	-	klasyfikator z siecią neuronową (zadana dwoma macierzami wag)
ann_training.m	-	kombajn tworzący sieć i uczący ją wsteczną propagacją błędów
backprop.m	-	implementacja jednej epoki uczenia stochastycznego wsteczną propagacją błędu
compErrors.m	-	błędy na podstawie macierzy pomyłek
confMx.m	-	macierz pomyłek z wyników klasyfikacji i prawdziwych etykiet
crann.m	-	tworzenie sieci neuronowej
mainscript.m	-	skrypt do uruchomienia podstawowych funkcji
readmnist.m	-	czytanie obrazów i etykiet
readSets.m	-	pojemnik na nazwy plików do rozpoznawania
tiny.txt	-	malutki zbiór danych do uruchomienia podstawowych funkcji

Na czerwono są zaznaczone funkcje, które z całą pewnością trzeba uzupełnić.