

# Podstawy uczenia maszynowego - Projekt 5

Damian Wasilenko, Dawid Macek; pn. 14:40, B

## 1 Cel projektu

Celem projektu jest porównanie trzech klasyfikatorów:

- AdaBoost bazującego na drzewach decyzyjnych
- Głębokiej sieci neuronowej bez warstw konwolucyjnych - MLP
- Głębokiej sieci neuronowej z warstwami konwolucyjnymi - CNN

## 2 Zbiór danych

- FMNIST
- obrazki 28x28
- $28*28=784$  cech
- 10 klas
- 70 tysięcy próbek.

## 3 Modele

### 3.1 AdaBoost

- Bazuje na drzewach decyzyjnych
- Przyjeliśmy głębokość drzewa równą 10

### 3.2 Multi Layer Perceptron - bez konwolucji

- Około 300 tysięcy parametrów
- Funkcja strat: **binary crossentropy**
- Metryka: **accuracy**
- Optimizer: **Adam**



Rysunek 1: Wizualizacja FMNIST

Typ	Parametr	Aktywacja
Wejście	28*28	-
Dense	64	Relu
Dropout	0.2	-
Dense	128	Relu
Dropout	0.2	-
Dense	256	Relu
Dropout	0.2	-
Dense	512	Relu
Dropout	0.2	-
Dense	256	Relu
Dropout	0.2	-
Wyjście	10	Softmax

Tablica 1: Architektura MLP

### 3.3 Convolutional Neural Network

- Około 300 tysięcy parametrów
- Funkcja strat: **binary crossentropy**
- Metryka: **accuracy**
- Optimizer: **Adam**

Typ	Opis	Kernel	Strides	Aktywacja
Wejście	(28, 28, 1)	-	-	-
Conv2D	32	(3, 3)	(1, 1)	Relu
BatchNormalization	-	-	-	-
Conv2D	32	(3, 3)	(1, 1)	Relu
BatchNormalization	-	-	-	-
Conv2D	32	(5, 5)	(2, 2)	-
BatchNormalization	-	-	-	-
Dropout	0.4	-	-	-
Conv2D	64	(3, 3)	(1, 1)	Relu
BatchNormalization	-	-	-	-
Conv2D	64	(3, 3)	(1, 1)	Relu
BatchNormalization	-	-	-	-
Conv2D	64	(5, 5)	(2, 2)	Relu
BatchNormalization	-	-	-	-
Dropout	0.4	-	-	-
Conv2D	128	(4, 4)	(1, 1)	Relu
BatchNormalization	-	-	-	-
Flatten	-	-	-	-
Dropout	0.4	-	-	-
Dense	10	-	-	Softmax

Tablica 2: Architektura CNN

## 4 Przebieg eksperymentów

Ze zbioru FMNIST wybierana jest pewien procent sampli. Domyślnie 50%, z wyjątkiem podpunktu **Jakość w zależności od ilości uczących**, w którym wybierane jest 10%, 30%, 50%, 100%.

Wybrany zbiór jest dzielony na zbiory: treningowy i testowy w stosunku 4:1. Potem następuje przetwarzanie zbioru treningowego (opcjonalne) np. zaszumianie. Następnie model jest trenowany z ustalonymi parametrami za pomocą przetworzonego zbioru treningowego i ewaluowany za pomocą testowego. Liczone są następujące metryki: `accuracy`, `f1 score`, `log loss`, `precision`, `recall`, `precision-recall auc`, `roc auc`.

### 4.1 Procedura zaszumiania danych treningowych

- Jako parametry przyjmuje procent danych treningowych do zaszumienia.
- Wyznacza ilość sampli do zaszumienia. `ilosc_do_zaszumienia = y_train.length*procent`
- Dla pierwszych `ilosc_do_zaszumienia` elementów w tablicy `y_train` ustawiamy element będący wynikiem `random.choice(classes)`

## 5 Jakość w zależności od ilości uczących

W ogólności im więcej danych tym lepsze wyniki [Rysunek 2].

Warte uwagi jest to, że modele ćwiczone dla dużej ilości danych szkolone były krócej (około 10 epok dla sieci) niż w kolejnym zadaniu. A mimo to najwyższy uzyskany wynik jest lepszy niż przy wydłużonym treningu.

AdaBoost zachowuje się dziwnie, ale może to wynikać ze zbyt małej liczby słabych klasyfikatorów (około 100).

## 6 Jakość klasyfikatorów w zależności od czasu treningu

Ze względu na to, że sieci trenujemy na karcie graficznej, a AdaBoosta na procesorze nie możemy porównać dokładnie obu metod. Wynika to z tego, że jeden model otrzymuje znacznie więcej mocy obliczeniowej w jednostce czasu.

Dlatego dla AdaBoosta mierzymy czas rzeczywisty, a dla sieci liczbę epok.

### 6.1 AdaBoost

Czas treningu jest zwiększany poprzez dodawanie kolejnych klasyfikatorów w grupie. Model wykazuje poprawę metryk wraz z czasem, ale widać tendencję do spłaszczania się [Rysunek 4]

### 6.2 Sieci neuronowe

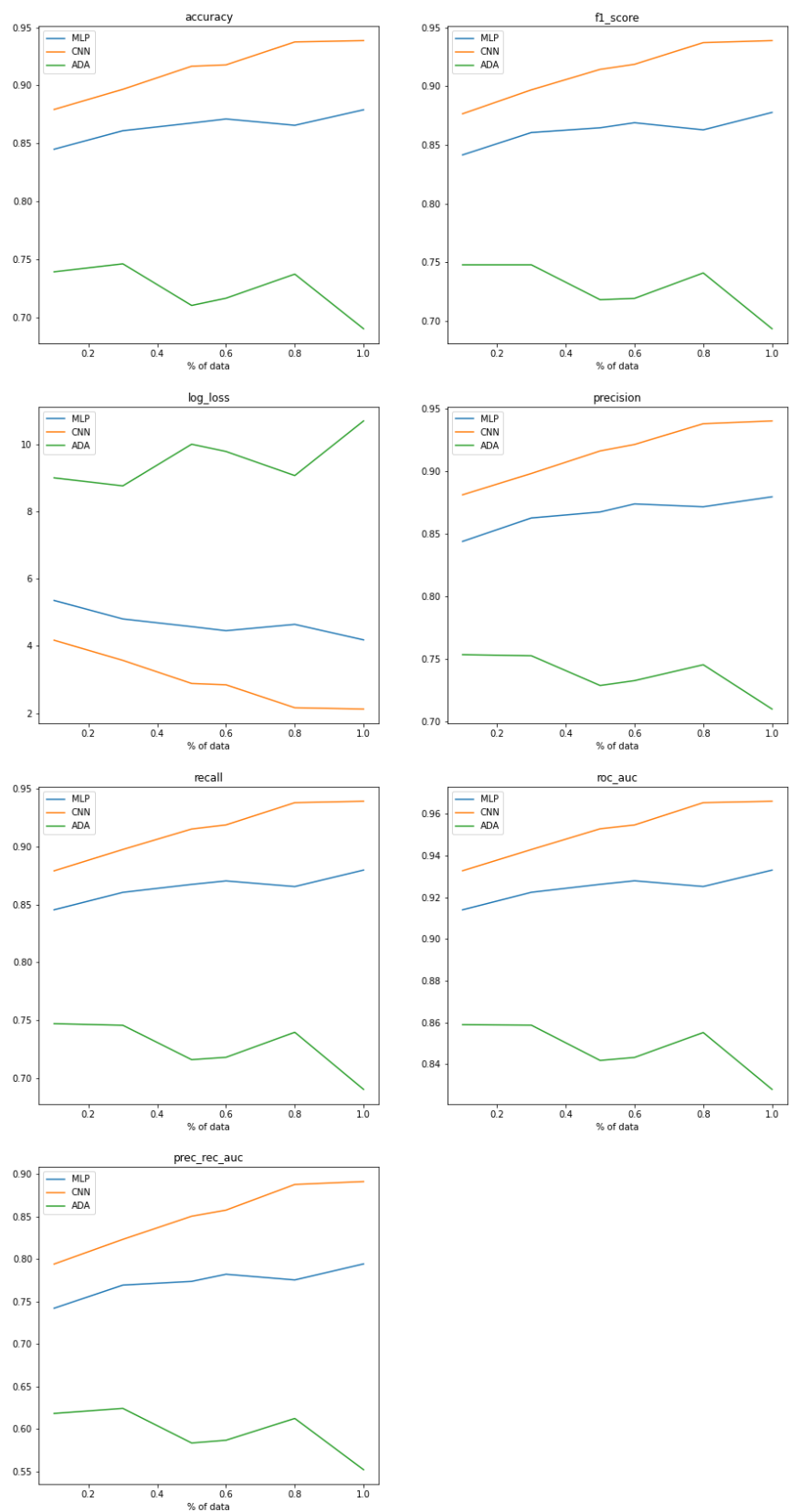
Sieci neuronowe także wydają się osiągać szczyt swoich możliwości od pewnej liczby epok. Z tym, że ten szczyt jest znacznie wyżej niż dla AdaBoosta [Rysunek 3].

## 7 Ocena mocy klasyfikatorów

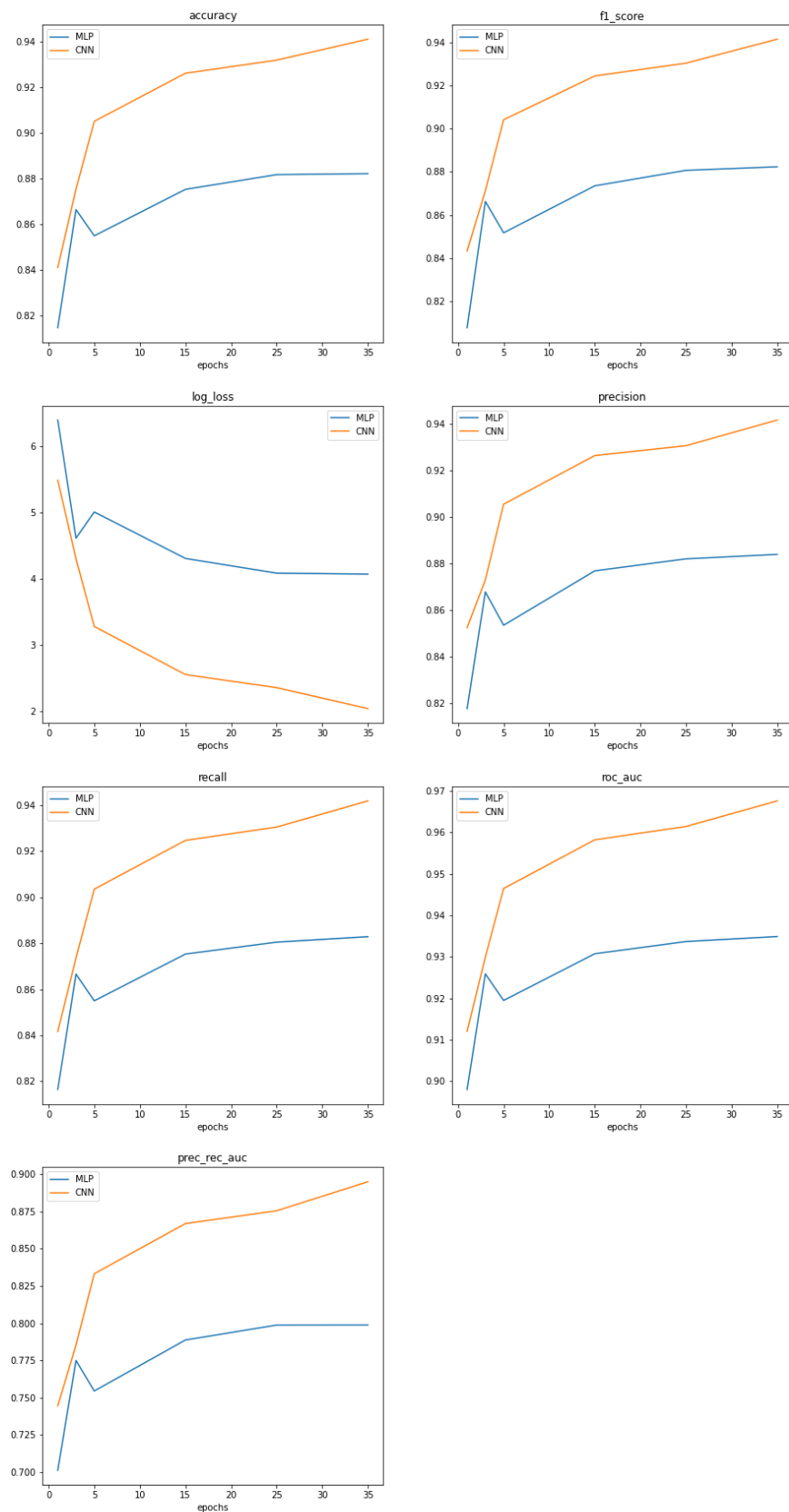
Wszystkie klasyfikatory tracą dokładność wraz ze zwiększaniem zaszumienia danych treningowych. Jedynym wyjątkiem jest klasyfikator MLP, w którym pojawiają się dziwne fluktuacje dokładności, ale może wynikać to z niestarannie dobranej architektury sieci. Finalnie każdy klasyfikator osiąga dokładność 10%, czyli staje się klasyfikatorem losowym [Rysunek 5].

## 8 Wnioski

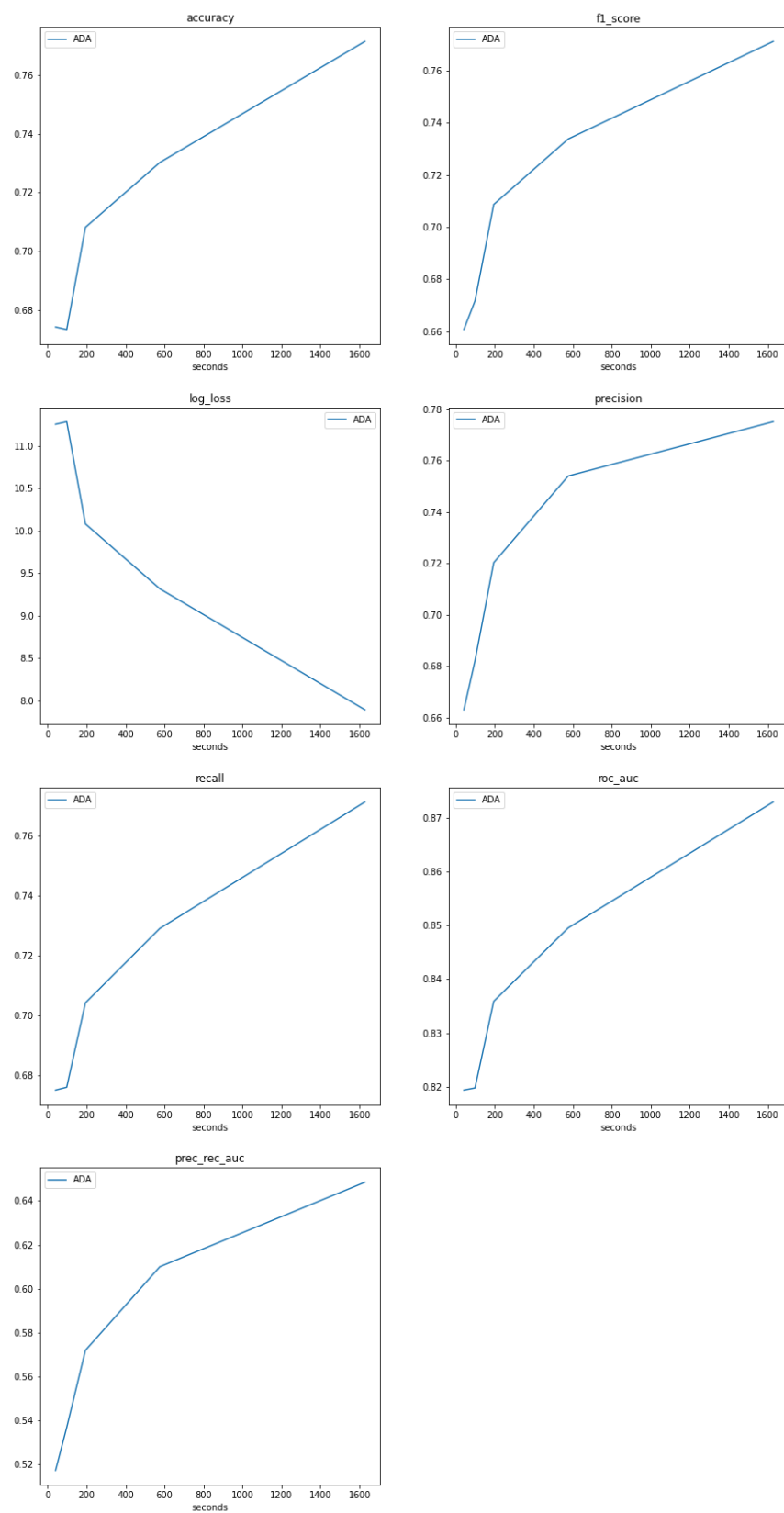
- Porównywanie metod szkolonych na różnych platformach sprzętowych jest trudne.
- Sieci neuronowe osiągają znacznie lepsze wyniki od AdaBoosta, ale nie można jednoznacznie stwierdzić, że to drugie jest gorsze ze względu na nierówności w platformach sprzętowych. AdaBoost poprawia się wraz ze zwiększaniem liczby słabych klasyfikatorów w zespole.
- Im więcej danych tym model daje lepsze rezultaty.
- Od pewnego momentu modele osiągają pewną dokładność, gdzie przedłużanie szkolenia nie daje żadnych rezultatów. W przypadku modeli o bardzo dużej liczbie parametrów i zbyt długim czasie treningu można doprowadzić do overfittingu.
- Zwiększanie ilości danych daje lepszy efekt niż zwiększanie długości trenowania.
- Wszystkie modele reagują podobnie na zaszumienie danych treningowych. Wraz ze zwiększaniem ilości niepoprawnych próbek treningowych zmniejsza się dokładność.



Rysunek 2: Miary jakości klasyfikatorów w zależności od ilości przykładów uczących.

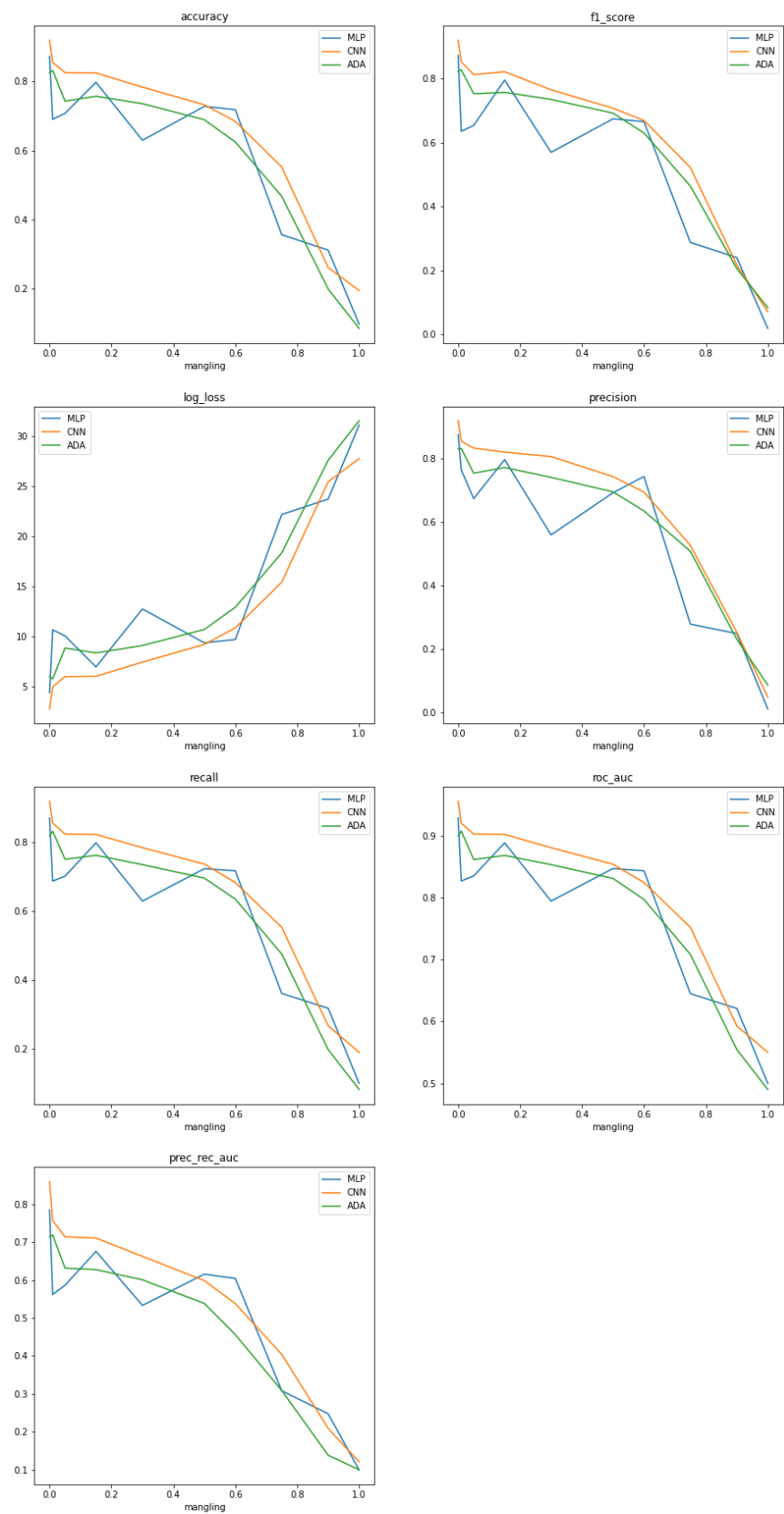


Rysunek 3: Miary jakości sieci w zależności od ilości epok



Rysunek 4: Miary jakości AdaBoosta w zależności od czasu treningu.





Rysunek 5: Miary jakości klasyfikatorów ze względu na stopień zaszumienia danych treningowych.