

Sieci Neuronowe, projekt II - Sieci Hopfielda

Filip Mieszkowski, Stanisław Kurzątkowski

November 2024

1 Wstęp

Celem tego projektu jest zaimplementowanie sieci Hopfielda, której zadaniem jest rozpoznawanie wzorców. Sieć Hopfielda jest jedną z najstarszych sieci neuronowych, została zaproponowana przez Johna Hopfielda w 1982 roku.

1.1 Budowa sieci

Sieć składa się z n neuronów, każdy z tych neuronów jest połączony z każdym innym neuronem wagą. Stan każdego neuronu może przyjmować tylko dwie wartości, -1 lub 1. Wagę pomiędzy neuronami i i j będziemy oznaczali jako w_{ij} . Wagi te są symetryczne, co oznacza, że $w_{ij} = w_{ji}$. Dodatkowo, waga w_{ii} jest równa 0, co oznacza, że neuron nie jest połączony z samym sobą. Wagi sieci będziemy przechowywać w macierzy wag W , gdzie $W = [w_{ij}]$. Z wymienionych powyżej własności wynika, że macierz wag jest symetryczna oraz posiada na przekątnej same zera.

1.2 Cel sieci

Sieć Hopfielda jest siecią asocjacyjną, co oznacza, że jest w stanie zapamiętać wiele wzorców i potrafi je rozpoznać, nawet jeśli są one zaszumione. Wzorce te są przechowywane w macierzy wag, która jest obliczana na podstawie wzorców uczących.

2 Działanie sieci

Działanie sieci Hopfielda można podzielić na dwa etapy: uczenie oraz rozpoznawanie wzorców.

2.1 Rozpoznawanie wzorców

Rozpoznawanie wzorców polega na prezentowaniu sieci wzorca, a następnie obserwacji zmian w stanie neuronów. Założmy, że wagi pomiędzy neuronami są ustalone. Pokazujemy sieci wzorec, który chcemy rozpoznać, tzn ustalamy

wartości neuronów na wartości z wzorca. Następnie, obserwujemy zmiany w stanie neuronów, aż do momentu, gdy sieć osiągnie stan stabilny. Dzieje się to w następujący sposób: wybieramy i -ty neuron, a następnie obliczamy sumę ważoną wejść do tego neuronu, czyli:

$$h_i = \sum_{j=1}^n w_{ij} x_j$$

gdzie x_j to stan j -tego neuronu (ponownie przyjmujemy $w_{ii} = 0$). W zależności od wartości h_i neuron przyjmuje wartość 1 lub -1, tzn jeżeli $h_i > b_i$ to $x_i = 1$, w przeciwnym wypadku $x_i = -1$, gdzie b_i to próg aktywacji neuronu i . Proces ten powtarzamy dla wszystkich neuronów, aż do momentu, gdy stan sieci się ustabilizuje. Możemy tutaj skorzystać z dwóch rozwiązań: asynchronicznego oraz synchronicznego. W rozwiązaniu asynchronicznym przeprowadzamy opisany wyżej proces dla każdego neuronu osobno, natomiast w rozwiązaniu synchronicznym dla wszystkich neuronów naraz.

2.2 Uczenie Hebbowskie

Uczenie sieci Hopfielda polega na obliczeniu macierzy wag W oraz progów aktywacji $B = [b_i]$ na podstawie wzorców uczących. Załóżmy, że mamy p wzorców uczących, które będziemy oznaczać jako $X = [x^{(1)}, x^{(2)}, \dots, x^{(p)}]$. Wówczas wagi ustalamy zgodnie ze wzorem:

$$w_{ij} = \frac{1}{p} \sum_{\mu=1}^p x_i^{(\mu)} x_j^{(\mu)}$$

zaś progi aktywacji ustalamy zgodnie ze wzorem:

$$b_i = \frac{1}{p} \sum_{\mu=1}^p x_i^{(\mu)}$$

gdzie $x_i^{(\mu)}$ oznacza i -ty neuron μ -tego wzorca uczącego. Warto zauważyć, że wagi są symetryczne, a progi aktywacji są ustalane na podstawie średniej wartości neuronów w danym wzorcu.

2.3 Dlaczego uczenie Hebbowskie działa?

Zdefiniujmy funkcję energii sieci Hopfielda jako:

$$E = -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n w_{ij} x_i x_j + \sum_{i=1}^n b_i x_i$$

gdzie x_i to stan i -tego neuronu. Możemy zauważyć, że funkcja ta jest zdefiniowana w taki sposób, że dla każdego wzorca uczącego $x^{(\mu)}$ jest minimum lokalne.

Dodatkowo zauważmy, że pochodna funkcji energii po x_i jest równa:

$$\frac{\partial E}{\partial x_i} = - \sum_{j=1}^n w_{ij} x_j + b_i$$

Zatem zmieniając wartość neuronu i -tego w kierunku $-\frac{\partial E}{\partial x_i}$ powinna zmniejszać się energia sieci. Nie zawsze musi się tak dziać, bowiem zmiana nawet w dobrym kierunku nie następuje w sposób ciągły, a od razu o dwie jednostki, co może okazać się "zbyt dużym krokiem". Oczekujemy jednak, że po wielu iteracjach energia sieci będzie maleć, co w końcu doprowadzi do osiągnięcia stanu stabilnego, który jest minimum lokalnym funkcji energii.

2.4 Falszywe minima lokalne: uczenie Oja

Załóżmy, że sieć Hopfielda dotrze po pewnym czasie do minimum funkcji energii. Niestety nie mamy żadnej pewności, że jest to minimum związane z jednym z wzorców uczących. Jeżeli wzorców uczących było więcej niż 1, to oprócz minimów związanych ze wzorcami uczącymi mogą istnieć również inne minima lokalne, które nie są związane z żadnym z wzorców uczących. W takim przypadku sieć może nie rozpoznać żadnego z wzorców uczących, a zamiast tego zbiegnie do jednego z fałszywych minimów lokalnych. Aby temu zapobiec, można skorzystać z uczenia Oja, które polega na dodaniu do reguły uczenia Hebbowskiego czynnika korygującego. Po zastosowaniu uczenia Hebbowskiego, wagi sieci są modyfikowane zgodnie ze wzorem:

$$\Delta w_{ij} = \frac{\eta}{n} \sum_{s=1}^n y_i^{(s)} (x_j^{(s)} - y_i^{(s)} w_{ij})$$

gdzie $y_i = \sum_{j=1}^n w_{ij} x_j$. Oczywiście nie aktualizujemy wagi w_{ii} , gdyż chcemy, żeby została ona równa 0. Taka korekta jest dokonywana kilka razy, dla pewnego ustalonego η . W praktyce pomaga to uniknąć fałszywych minimów lokalnych, co zwiększa skuteczność sieci.

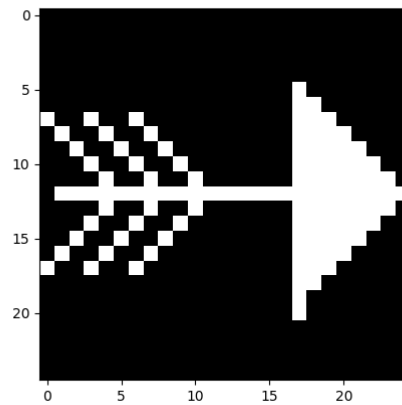
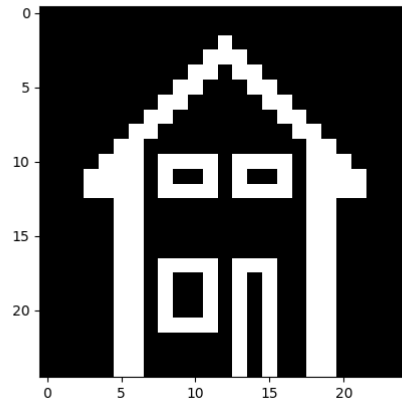
3 Eksperymenty

W ramach projektu przeprowadziliśmy kilka eksperymentów, które miały na celu zbadanie skuteczności sieci Hopfielda. Kod źródłowy eksperymentów 1 oraz 2 znajduje się w pliku *eksperyment.py*. W zależności od dobranej ziarna losowości, wyniki mogą się nieznacznie różnić.

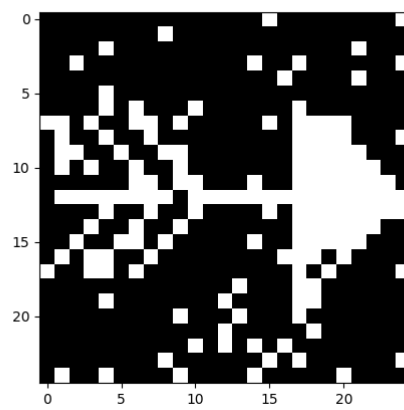
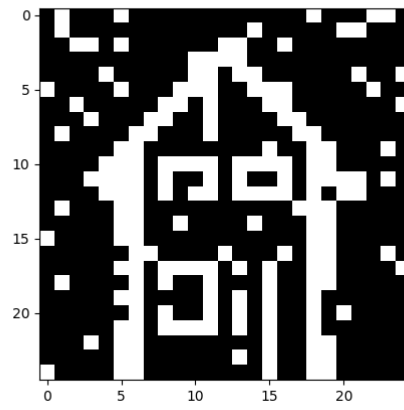
3.1 Eksperyment 1: rozpoznawanie wzorców zaszumionych

W pierwszym eksperymencie sprawdziliśmy, jak uczenie Oja wpływa na skuteczność sieci. W tym celu stworzyliśmy sieć Hopfielda, która miała za zadanie rozpoznawać wzorce zaszumione. Sieć o 625 neuronach została wyszkolona na 6 wzorcach, a następnie testowana na wzorcach zaszumionych. Zaszumienie polegało na zmianie 10% losowo wybranych pikseli na przeciwną wartość. Zbiorem

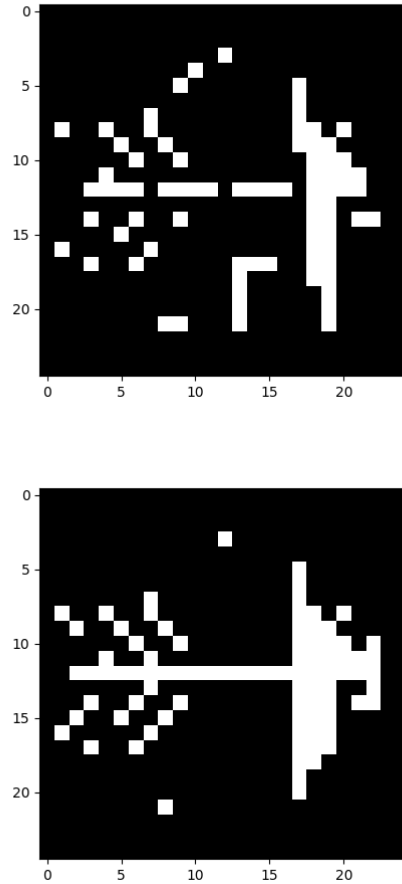
uczącym był zbiór *large* – 25×25 . Zwykłe uczenie Hebbowskie nie pozwalało na rozpoznanie wszystkich wzorców zaszumionych, dwa zbiegły do minimów lokalnych, które nie były związane z żadnym z wzorców uczących. Poniżej prezentujemy te dwa źle rozpoznane wzorce uczące:



Po zaszumieniu wzorce te wyglądały następująco:



Po zaprezentowaniu sieci zaszumionych wzorców, sieć nie była w stanie ich rozpoznać, zbiegła do następujących minimów lokalnych:



Szczególnie przy pierwszym z tych minimów lokalnych, można zauważyć, że jest on swgo rodzaju "połączeniem" obydwu wzorców uczących.

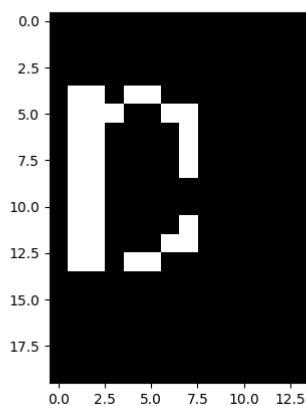
3.2 Eksperyment 2: rozpoznawanie wzorców zaszumionych z uczeniem Oja

W tym eksperymencie będziemy rozpoznawali dokładnie te same wzorce co w poprzednim eksperymencie, ale z użyciem uczenia Oja. Ustalmy liczbę iteracji uczenia Oja na 30. Zaś $\eta = 0.00001$. Jest to bardzo mała wartość, jednakże biorąc większą wartość η proces uczenia szybko rozbiegał do nieskończoności. Po zastosowaniu uczenia Oja, sieć była w stanie rozpoznać wszystkie wzorce zaszumione. Co również interesujące, po zastosowaniu uczenia Oja sieć zbiegała do do minimów lokalnych już w pierwszej iteracji (czyli po jednokrotnym zaktualizowaniu każdego neuronu), zaś w przypadku zwykłego uczenia Hebbowskiego

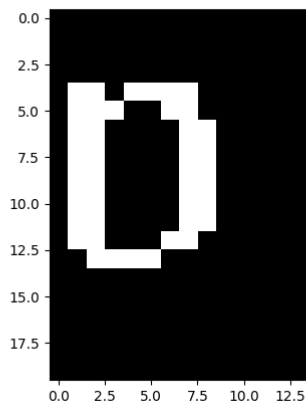
wzorzec, który zbiegł do niepoprawnego minimum lokalnego zbiegł do niego po trzech iteracjach.

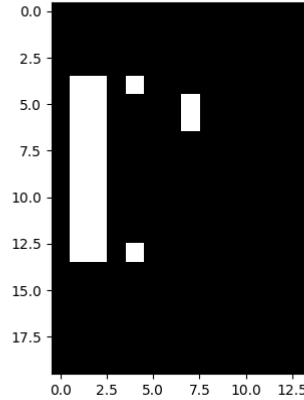
3.3 Eksperyment 3: rozpoznawanie liter alfabetu

W tym eksperymencie sprawdziliśmy, jak sieć Hopfielda radzi sobie z rozpoznawaniem liter alfabetu. W tym celu skorzystaliśmy z zbioru uczącego *letters* – 14×20 , który zawiera 26 liter alfabetu. Sieć została wyszkolona na 26 literach, a następnie testowana na literach zaszumionych. Zaszumienie polegało na zmianie 10% losowo wybranych pikseli na przeciwną wartość. Minimum lokalne, do którego zbiegła sieć dla wszystkich zaszumionych liter w przypadku uczenia Hebbowskiego wyglądało następująco:



Z kolei w przypadku uczenia Oja, sieć zbiegła do dwóch następujących minimów lokalnych:

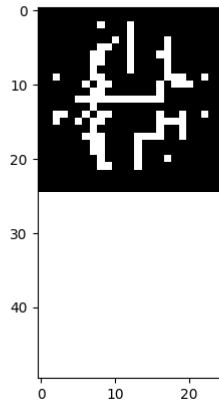




Jak widać, zarówno uczenie Hebbowskie jak i Oja nie pozwoliło na rozpoznanie liter zaszumionych. Prawdopodobnie wynika to z faktu, że obrazki liter alfabety są ze sobą silnie skorelowane, szczególnie, że w podanym zbiorze testowym wszystkie one znajdują się w lewym górnym rogu obrazka. Korelacja wzorców uczących stanowi dla sieci Hopfielda poważny problem, o czym można przeczytać na przykład w [1].

3.4 Eksperyment 4: porównanie podobnych zbiorów uczących

W tym eksperymencie nauczyliśmy sieć Hopfielda na zbiorze *large* – 25×25 oraz *large* – 25×50 . Są to niemalże te same wzorce, z tą różnicą, że w zbiorze *large* – 25×50 dolna połowa każdego obrazka jest cała biała. Wyniki uczenia na zbiorze *large* – 25×25 opisaliśmy w rozdziale 3.1. W przypadku uczenia na zbiorze *large* – 25×50 , sieć uczona metodą Hebbowską zupełnie sobie nie poradziła i zbiegła do minimów lokalnych, które nie były związane z żadnym z wzorców uczących, zaś wyglądały następująco:



Uczenie Oja dla odmiany pozwoliło na rozpoznanie wszystkich wzorców zaszumionych. Liczba iteracji równa była 30, zaś $\eta = 0.000002$ (zauważmy, że η jest mniejsza, niż w przypadku zbioru *large* – 25×25).

3.4.1 Wytłumaczenie wyników

Wyniki eksperymentów pokazują, że uczenie Oja jest znacznie bardziej skuteczne niż uczenie Hebbowskie. Rozważmy neuron należący do górnej części obrazka. W przypadku uczenia Hebbowskiego, neuron ten otrzymuje bardzo silny sygnał pochodzący z dolnej części obrazka (jest tam aż 625 neuronów, które zawsze mają tę samą wartość). W przypadku tak silnego sygnału z dolnej części obrazka neuron staje się niewrażliwy na stosunkowo słaby sygnał z górnej części obrazka. W przypadku uczenia Oja, wagi są zmniejszane, sygnał jest korygowany, co pozwala na lepsze rozpoznanie wzorców zaszumionych.

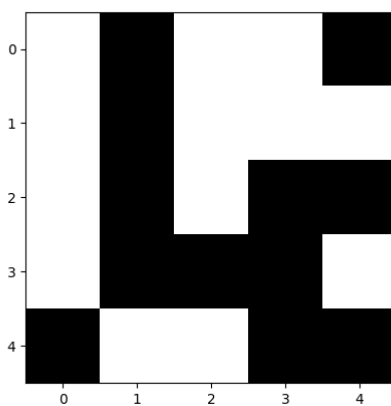
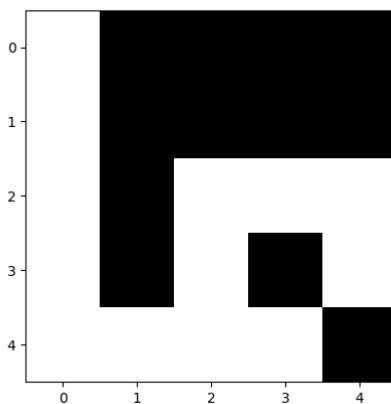
3.5 Eksperyment 5: poszukiwanie oscylatora

3.6 Eksperyment 6: duże obrazy

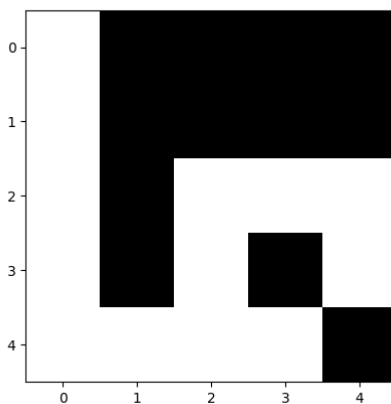
W tym eksperymencie sprawdziliśmy, jak sieć Hopfielda radzi sobie z dużymi obrazami. W tym celu wzięliśmy dwa obrazy kotów:



A następnie zamieniliśmy je na obrazy binarne:

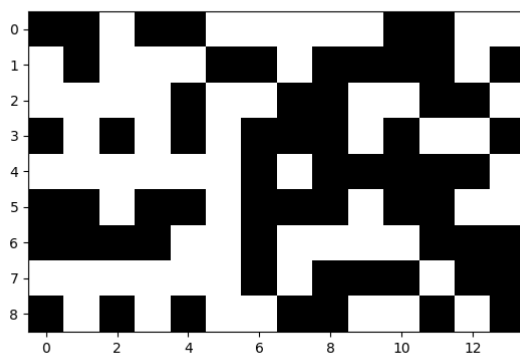


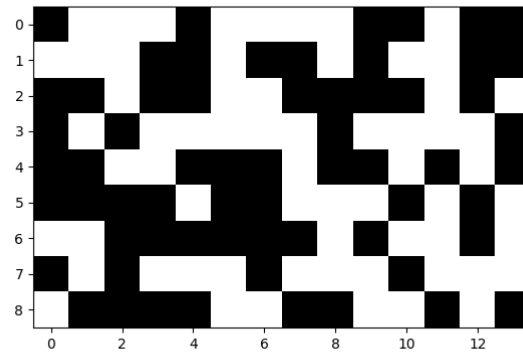
Na których następnie wyszkoliliśmy sieć Hopfielda metodą Hebbowską. Ze względu na dużą liczbę neuronów nie przeprowadzaliśmy uczenia Oja, które byłoby zbyt kosztowne obliczeniowo. Niestety, sieć nie była w stanie rozpoznać zaszumionych obrazów kotów. Po zaszumieniu obu obrazów na 10% sieć w obu przypadkach zbiegła do tego samego minimum lokalnego:



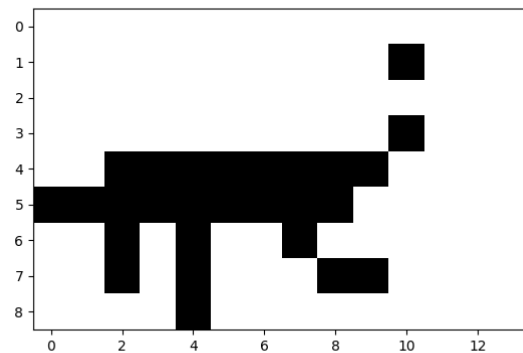
3.7 Eksperyment 7: Odzywskiwanie obrazu wyłącznie z szumu

Celem tego eksperymentu było sprawdzenie, jak sieć Hopfielda poradzi sobie z odzyskaniem obrazu z samego szumu. W tym celu przeprowadziliśmy uczenie na zbiorze *animals* – 14×9 , zarówno metodą Hebbowską jak i Oja. Następnie podaliśmy sieci obrazy, na których znajdował się wyłącznie szum:

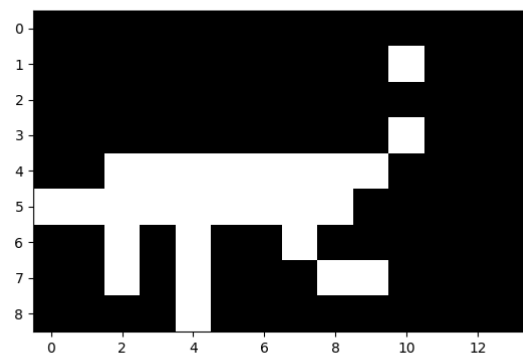




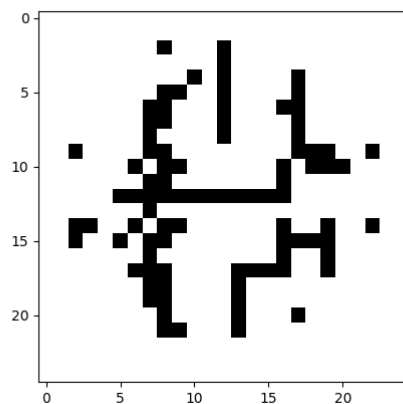
Wygenerowaliśmy 6 takich obrazów. Zarówno w przypadku uczenia metodą Hebbowską jak i Oja, w trzech przypadkach sieć zbiegła do następującego minimum lokalnego:

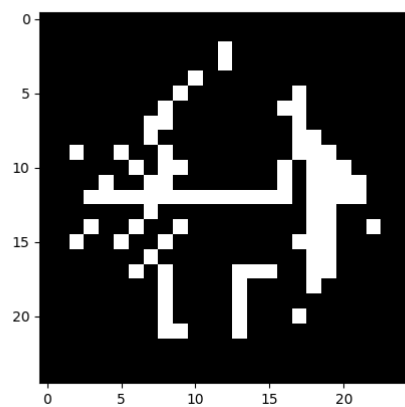
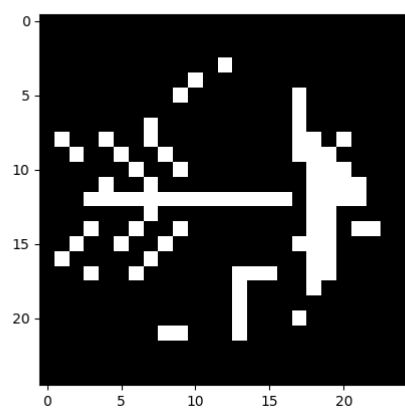
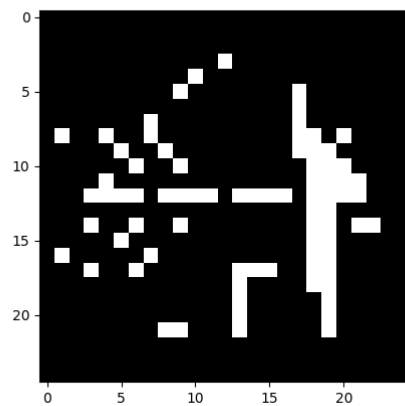


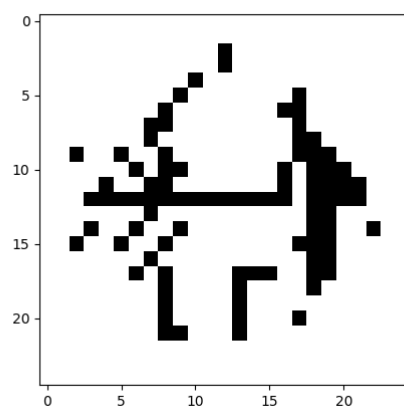
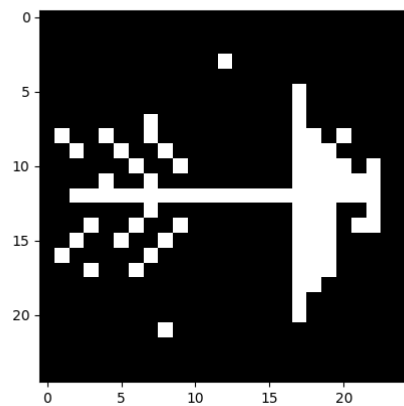
Zaś w pozostałych trzech przypadkach sieć zbiegła do następującego minimum lokalnego:



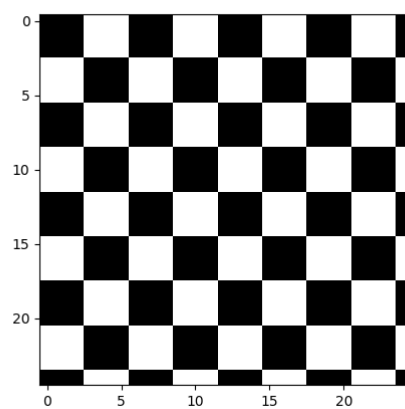
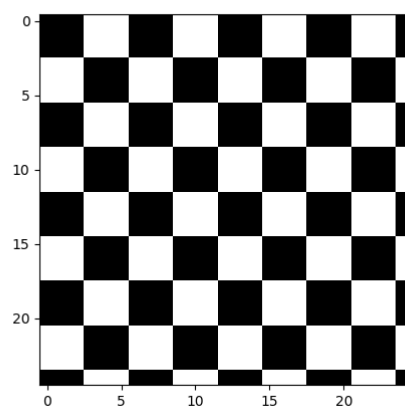
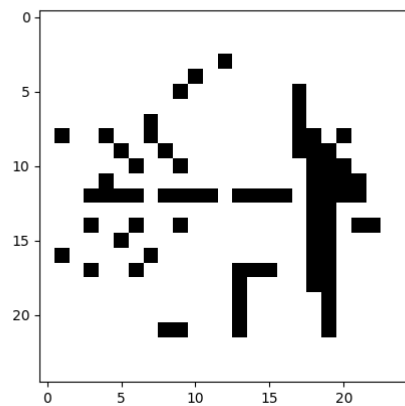
Widzimy zatem, że pomimo, że sieć nie była w stanie odzyskać obrazu z samego szumu, to zbiegła do w zasadzie tego samego minimum lokalnego, jednakże z przeciwnym znakiem. Co również ciekawe, minimum to wygląda jak połączenie wzorców uczących. Taki sam eksperyment przeprowadziliśmy dla zbioru *large-25x25*, uzyskując następujące wyniki przy uczeniu Hebbowskim:

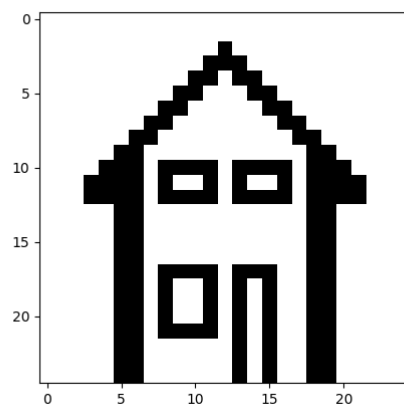
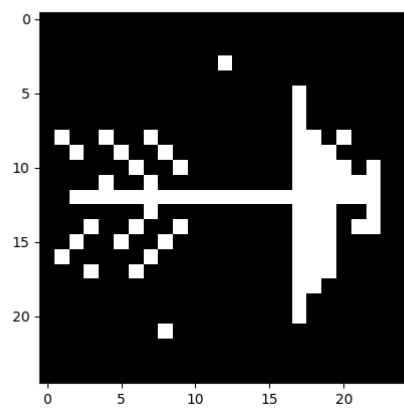
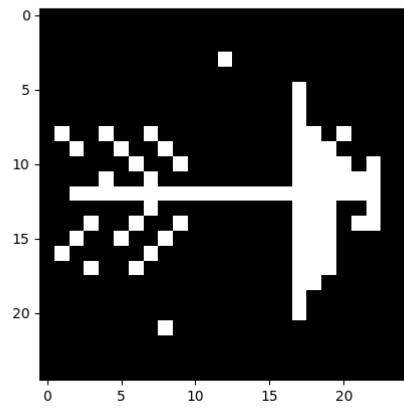






Zaś w przypadku uczenia metodą Oja:





Widzimy zatem, że uczenie Oja pozwoliło na lepsze odzyskanie obrazu z samego szumu, w porównaniu do uczenia Hebbowskiego.

References

- [1] Matthias Lowe *ON THE STORAGE CAPACITY OF HOPFIELD MODELS WITH CORRELATED PATTERNS*, The Annals of Applied Probability, 1998