

Eksperymenty z modelami do predykcji skupienia wzroku
przy użyciu głębszych sieci neuronowych

Karol Wójciak, Łukasz Zawadzki

11 stycznia 2024



EAIIIB / Katedra Informatyki Stosowanej
Akademia Górnictwo-Hutnicza im. Stanisława Staszica w Krakowie
Kraków, Polska

Streszczenie

W ramach niniejszej pracy naukowej skoncentrowaliśmy się na analizie i implementacji modeli predykcji skupienia wzroku przy użyciu głębokich sieci neuronowych. Przegląd literatury obejmuje różne podejścia i modele używane w tej dziedzinie, takie jak uczenie transferowe, sieci enkoder-dekoder, sieci odczytowe, sieci ensemble i sieci rekurencyjne. W pracy zaimplementowaliśmy dwa różne modele: enkoder-dekoder, U-Net, a także przeprowadziliśmy reimplementację modeli MSI-Net i DeepGaze II. Ewaluacja modeli została przeprowadzona na różnych zbiorach danych, a wyniki zostały porównane z istniejącymi benchmarkami.

Nasze eksperymenty pokazują, że mimo satysfakcjonujących rezultatów wizualnych, liczbowe wyniki skuteczności nie dorównują do rezultatów z prac opublikowanych na przestrzeni ostatnich lat. Wpływ na to miały ograniczenia czasowe, technologiczne oraz najprawdopodobniej źle dobrane parametry uczenia.

W celu poprawy wyników, zalecamy więcej eksperymentów z parametrami uczenia oraz wykorzystanie większych zbiorów danych. Mimo napotkanych trudności, praca dostarcza wglądu w różne podejścia do predykcji skupienia wzroku przy użyciu głębokich sieci neuronowych.

1 Wprowadzenie

Współcześnie rozwijająca się dziedzina sztucznej inteligencji i głębokiego uczenia maszynowego zdobywa coraz większe uznanie w dziedzinie analizy i interpretacji ludzkiego zachowania. Jednym z ciekawych obszarów badań jest predykcja skupienia wzroku, która pozwala zrozumieć, jak ludzie postrzegają i reagują na otaczający świat. Niniejsza praca naukowa skupia się na wykorzystaniu sieci neuronowych do predykcji skupienia wzroku.

W kontekście badań nad skupieniem wzroku, zdolność do precyzyjnego przewidywania obszarów, na które skierowana jest uwaga obserwatora, ma znaczące implikacje dla wielu dziedzin, takich jak projektowanie interfejsów użytkownika, badania marketingowe, psychologia poznawcza czy rozwijające się dziedziny związane z interakcją człowiek-maszyna. Wykorzystanie sieci neuronowych do analizy wzorców skupienia wzroku stwarza nowe możliwości w interpretowaniu i modelowaniu złożonych procesów percepcyjnych.

W kolejnych rozdziałach pracy omówimy najnowsze badania i osiągnięcia w dziedzinie, analizując różne podejścia i modele używane do tego celu. Dodatkowo, porównamy uzyskane przez nas wyniki z tymi osiąganyimi przez nowoczesne modele. Naszym celem jest dostarczenie kompleksowego spojrzenia na to, jak sieci neuronowe mogą efektywnie przewidywać obszary skupienia wzroku, co może przyczynić się do rozwoju nowoczesnych technologii oraz lepszego zrozumienia ludzkiej percepcji w kontekście interakcji z otoczeniem.

2 Przegląd literatury

W tej sekcji omówimy różne metody, modele, metryki i funkcje strat wykorzystywane w rozwiązywaniu problemu predykcji skupienia wzroku.

2.1 Uczenie transferowe

W związku z ogromną różnorodnością cech, a co za tym idzie, dużymi wymaganiami do ich uogólnienia, jakimi cechują się zbiory zdjęć, znaczna większość sieci neuronowych zajmujących się w jakiś sposób przetwarzaniem obrazów korzysta ze stworzonych wcześniej sieci *backbone*, które zostały wytrenowane na bardzo dużych zbiorach zdjęć, jak ImageNet. Sam zbiór ImageNet, zawiera ponad 14 milionów zdjęć, co daje solidną podstawę do uogólnienia wytrenowanych na nim sieci. Podejście takie pozwala na skuteczne trenowanie modelu nawet na relatywnie niewielkich zbiorach danych, co jest kluczowe ze względu na ograniczone dostępne dane treningowe w dziedzinie analizy skupienia wzroku.

Jednym z takich *backbone* jest zaproponowany przez w pracy [Simonyan and Zisserman, 2014] VGG16, oraz jego wariacja VGG19. Jest to sieć konwolucyjna w formie enkodera, złożona z 5 bloków, po kilka warstw każda. Inne znane *backbones* to m.in Xception, AlexNet, GoogLeNet, ResNet-34.

2.2 Enkoder-dekoder

Klasycznym i najstarszym podejściem do tego problemu z użyciem sieci neuronowych jest zastosowanie sieci typu enkoder-dekoder. Zazwyczaj opierają się one na wytrenowanym wcześniej enkoderze, typu VGG16/VGG19, którego wagi nie podlegają dalszemu uczeniu. Następnie wynik otrzymany z tej części sieci jest przetwarzany przez dekoder, którego architektura różni się w zależności od podejścia do problemu.

Jednym z niestandardowych podejść jest to zaproponowane przez w pracy [Kroner et al., 2020], które polega na przekazaniu do specjalnie zaprojektowanej warstwy ASPP (*Atrous Spatial Pyramid Pooling*) dekodera zebranych wyników kilku warstw enkodera. Autorzy wskazują, że pozwala to na zebranie różnych zbiorów cech, powstałych podczas przetwarzania obrazu w enkoderze, co wpływa pozytywnie na ogólne wyniki sieci.

2.3 Sieci odczytowe

W pracach [Kümmerer et al., 2016], [Linardos et al., 2021], [Kümmerer et al., 2022] autorzy proponują wykorzystanie uczenia transferowego (*transfer learning*) wykorzystując różne sieci typu enkoder przeuczone na zbiorze ImageNet. Dodatkowo, autorzy w swoich modelach DeepGaze pozwalają na douczanie kilku ostatnich warstw z sieci enkodera, za to ich część "dekodująca" (nazwana przez autorów siecią odczytową - *readout network*) jest bardzo mała, jedynie kilka warstw konwolucyjnych o rozmiarze jądra 1x1. Oprócz tego wprowadzany jest element architektury modelu nazwany jako *center bias*, czyli macierz z wagami, dzięki której symulowana jest rzeczywista skłonność ludzi do patrzenia się na elementy bliżej środka obrazu, jak pokazano w pracach [Buswell, 1935], [Tatler, 2007], [Kienzle et al., 2009].

Podobne rozwiązanie proponują autorzy modelu ML-Net [Cornia et al., 2016]. Wykorzystują oni jedynie dwie warstwy konwolucyjne po części *backbone* oraz również korzystają z *center bias*.

W modelu DeepGaze III [Kümmerer et al., 2022] autorzy dodają osobną podsieć odpowiedzialną za uczenie się ścieżek patrzenia na kolejne elementy obrazu.

2.4 Sieci ensemble

Innym podejściem do tego problemu jest wykorzystanie ensemble. Używane jest ono w modelu DeepGaze IIIE [Linardos et al., 2021] w postaci pięciu różnych *backbone*: DenseNet-201, ResNext-50, EfficientNet-B5, ShapeNet-C, ResNet-50. Rezultaty z każdego z nich są następnie scalane w jeden obraz wyjściowy, dzięki czemu uzyskiwane są lepsze wyniki.

Jeszcze inne podejście zaproponowali autorzy pracy [Hachaj et al., 2021], gdzie z kolejnych poziomów enkodera są połączenia do oddzielnych dekoderów, a następnie ich wyniki są łączone w jeden obraz wyjściowy.

2.5 Sieci rekurencyjne

Kolejnym podejściem, raczej rzadko stosowanym do predykcji skupienia wzroku, są sieci rekurencyjne. Pracą, która w tym kontekście szczególnie zwróciła naszą uwagę, jest [Ding et al., 2022], w której autorzy zaproponowali innowacyjne połączenie elementów sieci rekurencyjnej oraz zrównolegionych warstw konwolucyjnych, zarówno w obrębie enkodera jak i dekodera. Warto zaznaczyć, że wytrenowana przez autorów sieć SalFBNet plasuje się jako jedna z najlepszych według zestawienia *MIT/Tuebingen Saliency Benchmark* [Kümmerer et al., 2018].

2.6 Metryki i funkcje straty

Dla modeli wykorzystujących sieci neuronowe równie ważne jak sama architektura sieci jest dobranie odpowiednich funkcji strat (*loss*), które znaczco wpływają na proces uczenia - działanie modelu nauczonego z jedną funkcją straty często jest zupełnie inne od działania tego samego modelu z inną funkcją straty, niekiedy nawet jeden nie nadaje się do danego problemu, gdy drugi bardzo dobrze sobie radzi.

Dlatego też, w przypadku predykcji skupienia wzroku, standardowymi funkcjami straty są: NSS, KLD lub rzadziej MSE, jak opisano w pracy [Kummerer et al., 2018].

Trzeba odnotować, że funkcja KLD zaimplementowana w bibliotece TensorFlow jest obliczana piksel po pikselu, co nie jest przydatne w kontekście przewidywania skupienia wzroku. Dlatego, idąc za [Kroner et al., 2020], nasza zmodyfikowana funkcja straty KLD wygląda następująco:

$$D_{KL}(P||Q) = \sum_i Q_i \ln(\epsilon + \frac{Q_i}{\epsilon + P_i}) \quad (1)$$

Jednak jak zauważają autorzy SalFBNet [Ding et al., 2022], NSS bierze pod uwagę tylko punkty ze skupieniem wzroku, pomijając obszary bez skupienia wzroku, czyli nie "kara" modelu za nadmiarowe przewidywania. Dlatego zaproponowali oni swoją funkcję sFNE (*Selective Fixation and Non-Fixation Error*), która, wybierając losowe punkty z obrazu, ma zaradzić temu problemowi.

Przy uczeniu, i co ważniejsze, ewaluacji sieci, wykorzystuje się również metryki, które co prawda nie mają bezpośredniego wpływu na proces uczenia, jednak służą porównaniu ze sobą różnych modeli. Autorzy pracy [Kummerer et al., 2018] wskazują i opisują kilka najważniejszych - AUC i sAUC (*(Shuffled) Area Under Curve*), IG (*Information Gain*), CC (*Correlation Coefficient*) oraz SIM (*Similarity*).

3 Zakres i cel prowadzonej pracy

Prowadzony przez nas projekt miał głównie na celu:

- Zaznajomienie z technologiami związanymi z predykcją skupienia wzroku, w tym głębokimi sieciami neuronowymi, narzędziami programistycznymi i frameworkami używanymi w tej dziedzinie.
- Próbę odtworzenia istniejących modeli predykcji skupienia wzroku, opisanych w kilku klu- czowych pracach naukowych, celem lepszego zrozumienia ich architektury, funkcji straty oraz ogólnej skuteczności.
- Próbę stworzenia własnych modeli predykcji skupienia wzroku, wykorzystując zdobytą wcze- śniej wiedzę i analizę istniejących prac.
- Eksperymentowanie z różnymi architekturami i parametrami modeli w celu zbadania ich wpływu na skuteczność predykcji.
- Porównanie stworzonych modeli z dostępnymi benchmarkami w dziedzinie predykcji sku- pienia wzroku.

4 Zbiory danych i zastosowana metodologia

4.1 Zbiory danych

W tabeli 1 przedstawiliśmy wszystkie zbiory użyte przez nas podczas procesu uczenia i walidacji modeli. Należy odnotować, że ostatnie 4 zbiory są częścią większego zbioru PseudoSaliency stwo- rzonego przez autorów modelu SalFBNet poprzez poszerzenie innych, ogólnie dostępnych zbiorów (*data augmentation*) za pomocą m.in omawianych wcześniej modeli - DeepGaze II, MSINet. Wynikowy zbiór liczy aż 150 000 rekordów, jednak my ograniczyliśmy się do użycia tylko kilku załączonych podzbiorów. Podczas ewaluacji modeli użyliśmy zbioru MIT1003.

Lp.	Nazwa zbioru	Liczba rekordów	Link
1	SALICON	15000	[Jiang et al., 2015]
2	DUT-OMRON	5168	[Yang et al., 2013]
3	OSIE	700	[Xu et al., 2014]
4	MIT1003	1003	[Judd et al., 2009]
5	CSSD	200	[Ding et al., 2022]
6	ECSSD	1000	[Ding et al., 2022]
7	MSRA10K	10000	[Ding et al., 2022]
8	MSRA-B	5000	[Ding et al., 2022]

Tabela 1: Zbiory danych

4.2 Użyte technologie

Przy implementacji wszystkich naszych modeli oraz reimplementacji modeli istniejących użyliśmy środowiska Python z biblioteką TensorFlow i Keras. Aby przyspieszyć obliczenia, skorzystaliśmy z biblioteki tensorflow-directml-plugin, umożliwiającej wykorzystanie karty graficznej RX 5700 XT w procesie uczenia sieci. Jednocześnie korzystaliśmy również z usługi Google Colaboratory, która za darmo udostępnia możliwość prowadzenia obliczeń na serwerze online wyposażonym w kartę graficzną Nvidia Tesla T4.

Tam, gdzie było to wymagane, do obróbki zdjęć (m.in. zmiany rozmiaru, ale też przy wyświetlaniu) użyliśmy biblioteki OpenCV.

4.3 Metodologia

Przeprowadziliśmy reimplementację wybranych istniejących modeli predykcji skupienia wzroku na najnowsze wersje bibliotek lub, gdy model był zaimplementowany w innym środowisku (np. PyTorch), zaadaptowaliśmy go w TensorFlow. Ten krok pozwolił na dostosowanie architektury modeli do ewentualnych zmian w API oraz zoptymalizowanie ich działania w kontekście najnowszych rozwiązań.

Przy niektórych modelach eksperymentowaliśmy z lekko zmienionymi architekturami lub innymi parametrami uczenia, niż oryginalnie zaproponowanymi. Poddaliśmy modele procesowi uczenia na różnych zbiorach danych. Po każdym etapie uczenia, przeprowadziliśmy testy skuteczności modeli na zbiorach testowych. Dodatkowo, przetestowaliśmy te same modele z różnymi konfiguracjami parametrów - szybkość uczenia (*learning rate*), funkcja straty (*loss function*), optymalizator rozwiązania (*optimizer*).

5 Rezultaty

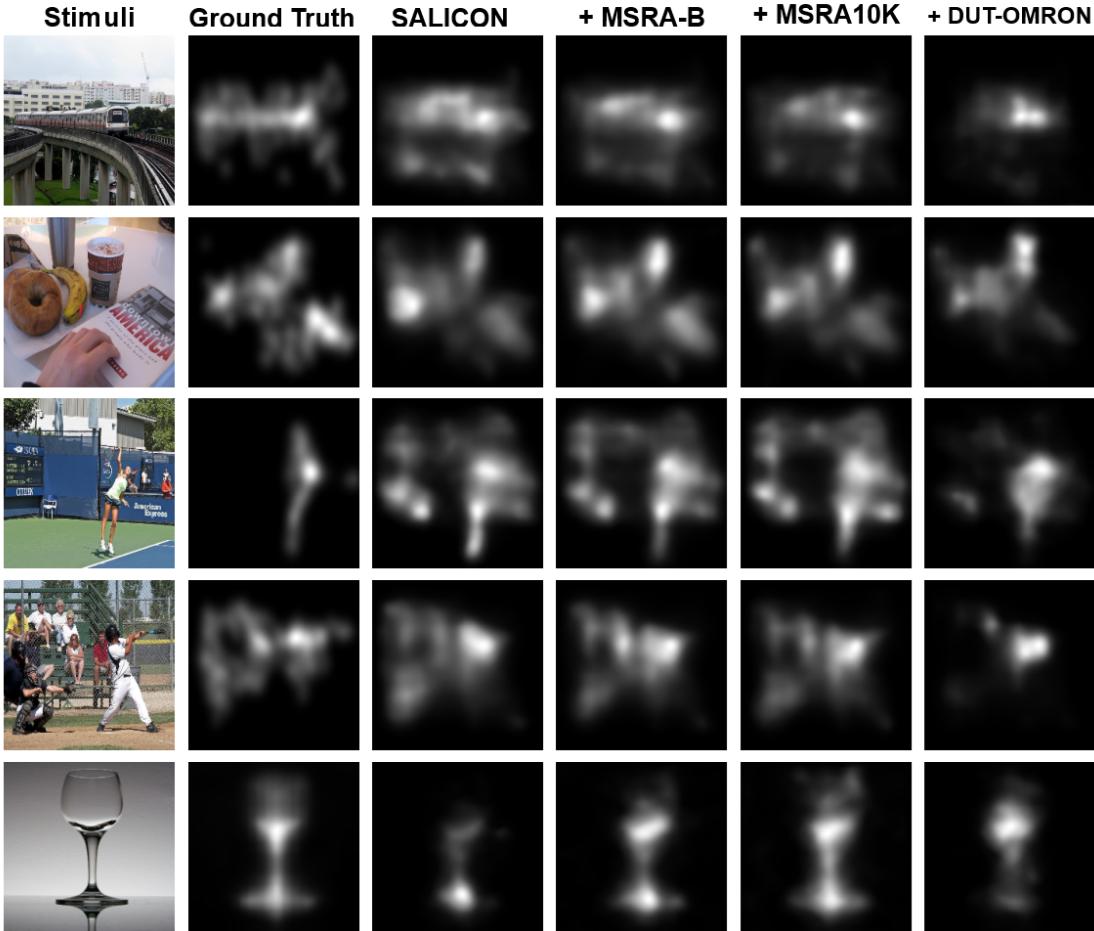
W sekcjach poniżej opiszemy zaprojektowane przez nas modele i uzyskane przez nie wyniki podczas ewaluacji. Porównamy je z naszą implementacją modelu MSI-NET oraz DeepGaze II.

5.1 Enkoder-dekoder

Sieć o klasycznej architekturze enkoder-dekoder. Funkcję enkodera pełni *backbone* VGG19 z wagami ImageNet, natomiast dekoder składa się z 5 bloków, z których pierwsze dwa składają się z warstwy *Upsampling* i dwóch warstw *Conv2D* (*ReLU*, "same", 3x3), a pozostałe z *Upsampling* + 3 warstwy *Conv2D*. W każdym bloku warstwy mają tyle samo filtrów, a ich liczba zmniejsza się o połowę pomiędzy blokami, z początkowych 256 do 16 w przedostatniej warstwie i 1 filtra w warstwie wyjściowej, by uzyskać czarno-białe (jedno-kanałowe) zdjęcie. Jako optymalizatora użyliśmy Adam z *learning rate* = 0.00001 (1e-5), oraz MSE jako funkcji straty.

Zbiory	AUC \uparrow	KLD \downarrow	CC \uparrow	SIM \uparrow
SALICON	0.731	1.856	0.250	0.311
+ MSRA-B	0.742	1.752	0.274	0.321
+ MSRA10K	0.745	1.765	0.265	0.319
+ DUT-OMRON	0.735	1.907	0.269	0.319

Tabela 2: Wyniki ilościowe modeli - w kolejnych wierszach model dotrenowany na kolejnym podanym zbiorze, tj. w ostatnim wierszu wytrenowany na wszystkich podanych zbiorach.



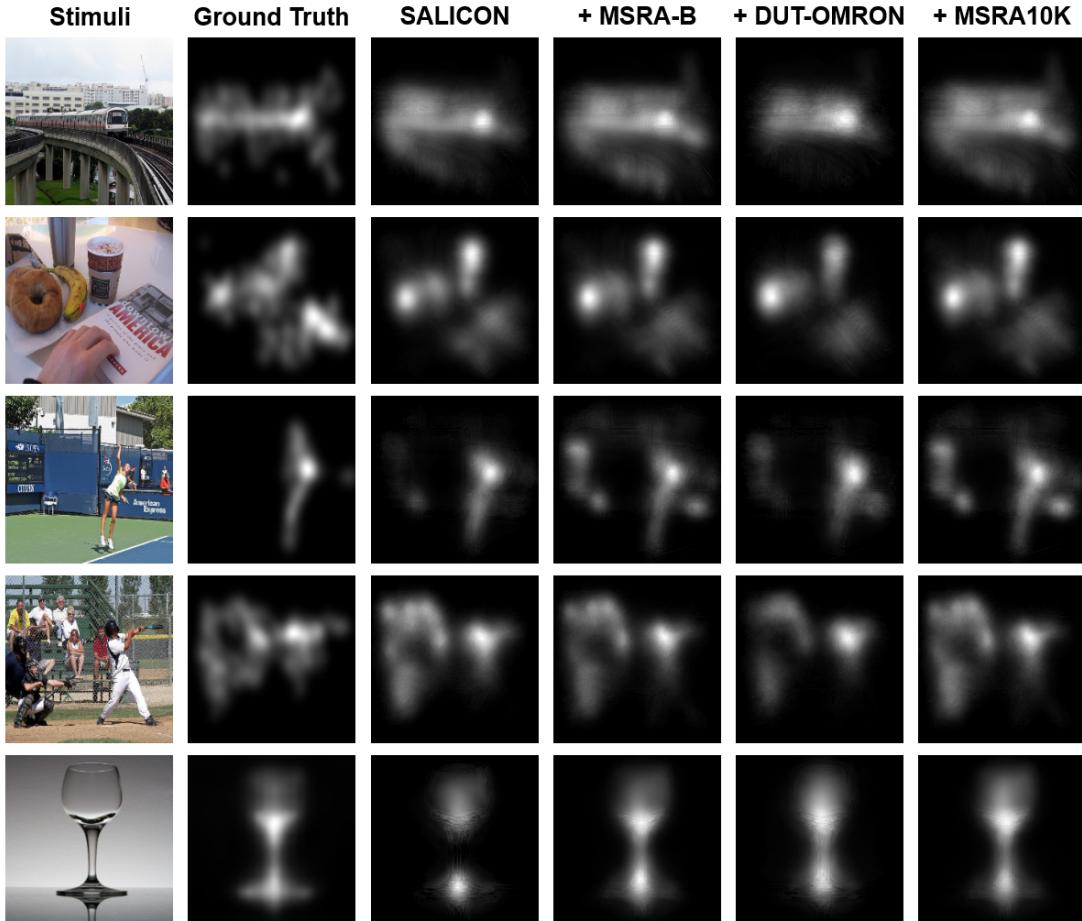
Rysunek 1: Przykładowe wyniki modeli Enkoder-dekoder

5.2 U-Net

Sieć o architekturze zbliżonej do opisanego powyżej enkodera-dekodera. Stworzyliśmy modele w dwóch wariantach: w pierwszym - podobnie jak w [Ronneberger et al., 2015], stworzyliśmy sieć opartą o *backbone* VGG19, w której pomiędzy wszystkimi 5 blokami enkodera i dekodera na tych samych poziomach są połączenia między warstwami. W drugim - usunęliśmy zewnętrzne dwa połączenia oraz zmniejszyliśmy rozmiary bloków z 3 do dwóch warstw *Conv2D*.

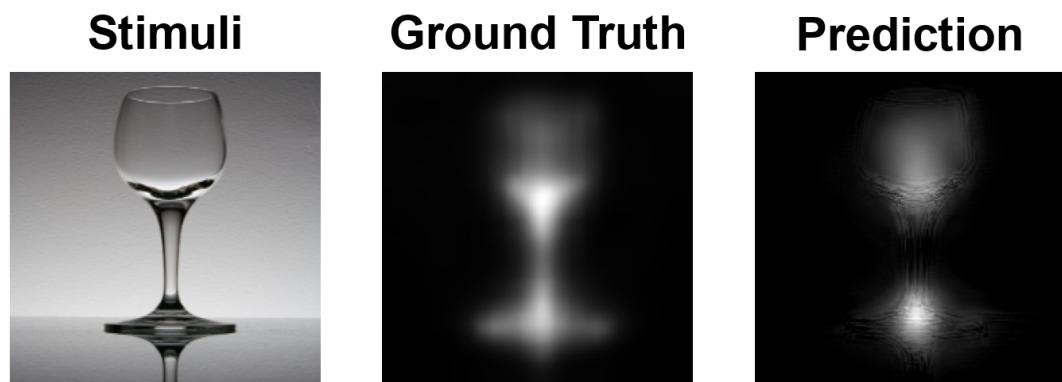
Zbiory	AUC \uparrow	KLD \downarrow	CC \uparrow	SIM \uparrow
SALICON	0.663	2.203	0.219	0.294
+ MSRA-B	0.646	2.002	0.230	0.298
+ DUT-OMRON	0.699	2.215	0.228	0.300
+ MSRA10K	0.645	1.922	0.235	0.301

Tabela 3: Wyniki ilościowe modeli - w kolejnych wierszach model dotrenowany na kolejnym podanym zbiorze, tj. w ostatnim wierszu wytrenowany na wszystkich podanych zbiorach.

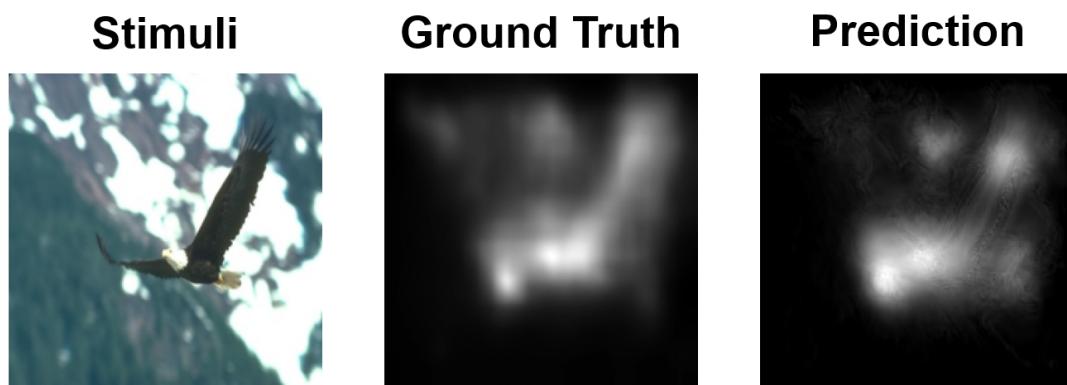


Rysunek 2: Przykładowe wyniki modeli U-Net

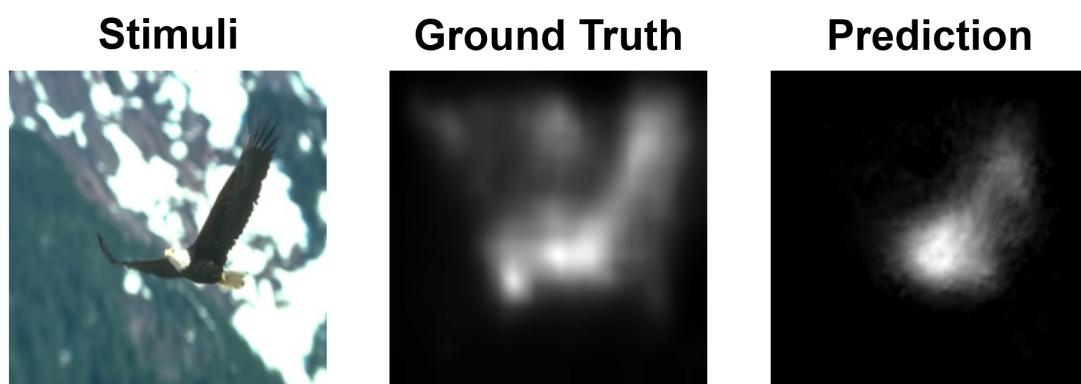
Warto zauważać specyfikę sieci U-Net na poniższych przykładach - [3](#) oraz [4](#). W obu przypadkach widać, że oryginalne zdjęcie "prześwitują" w wyjściu uzyskanym z sieci. Przypuszczamy, że jest to wynik połączeń pomiędzy warstwami enkodera i dekodera - powodują one zebranie wysokopoziomowych, nie przetworzonych jeszcze cech, co potem przejawia się jako właściwe prześwytywanie na wynikowym obrazie.



Rysunek 3: Przykład "prześwitywania" widoczny na ostatnim obrazie



Rysunek 4: Przykład "prześwitywania" widoczny na ostatnim obrazie



Rysunek 5: Rezultat zredukowanej sieci U-Net. Widać rozpiselizowanie wynikowego obrazu.

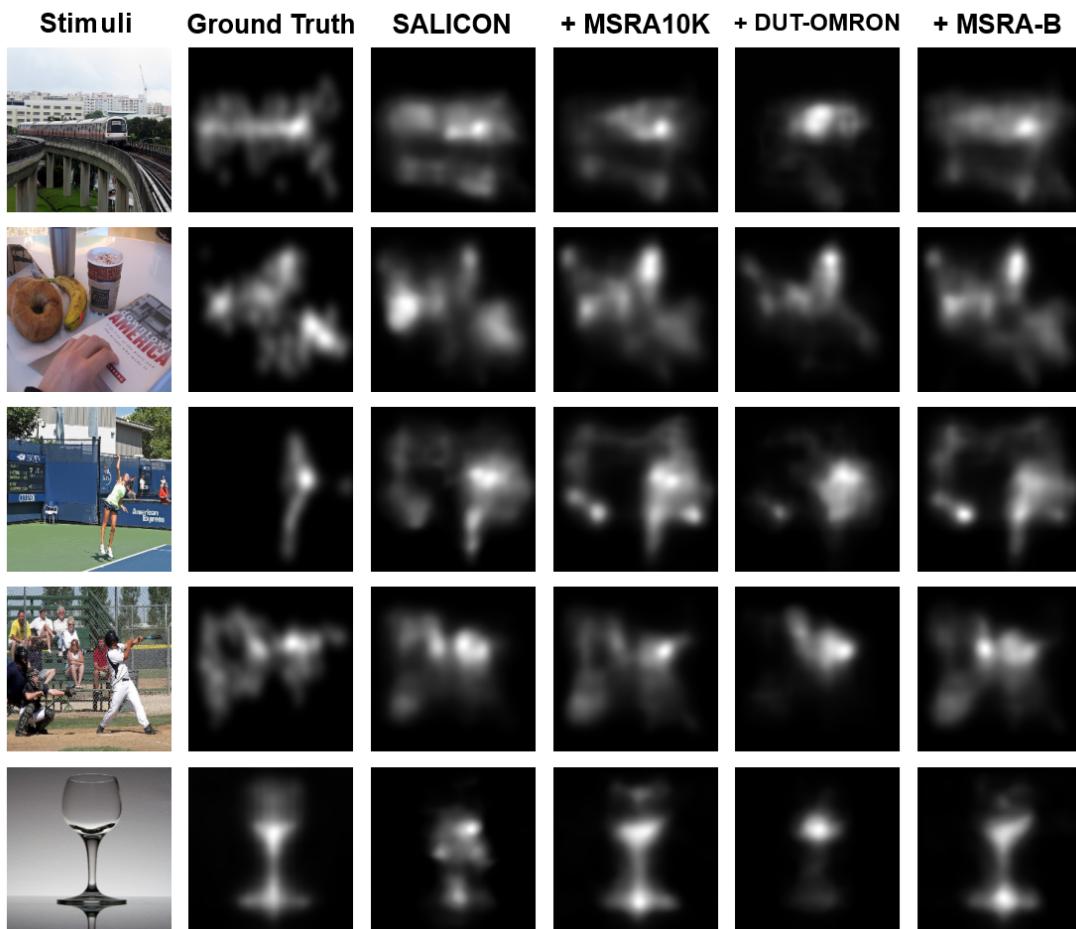
Graficzne wyniki modelu z usuniętymi dwoma połączeniami nie były zadowalające, jak można zauważyć na 5, dlatego postanowiliśmy nie uwzględnić tabelki z metrykami.

5.3 MSI-Net

Reimplementacja modelu z pracy [Kroner et al., 2020]. Użyliśmy optymalizatora Adam z *learning rate* wynoszącym 0.00001 oraz funkcję straty KLD (zaimplementowaną zgodnie z równaniem 1).

Zbiory	AUC ↑	KLD ↓	CC ↑	SIM ↑
SALICON	0.729	1.899	0.251	0.310
+ MSRA10K	0.740	1.799	0.267	0.318
+ DUT-OMRON	0.744	1.892	0.283	0.326
+ MSRA-B	0.734	1.817	0.253	0.312

Tabela 4: Wyniki ilościowe modeli - w kolejnych wierszach model dotrenowany na kolejnym podanym zbiorze, tj. w ostatnim wierszu wytrenowany na wszystkich podanych zbiorach.



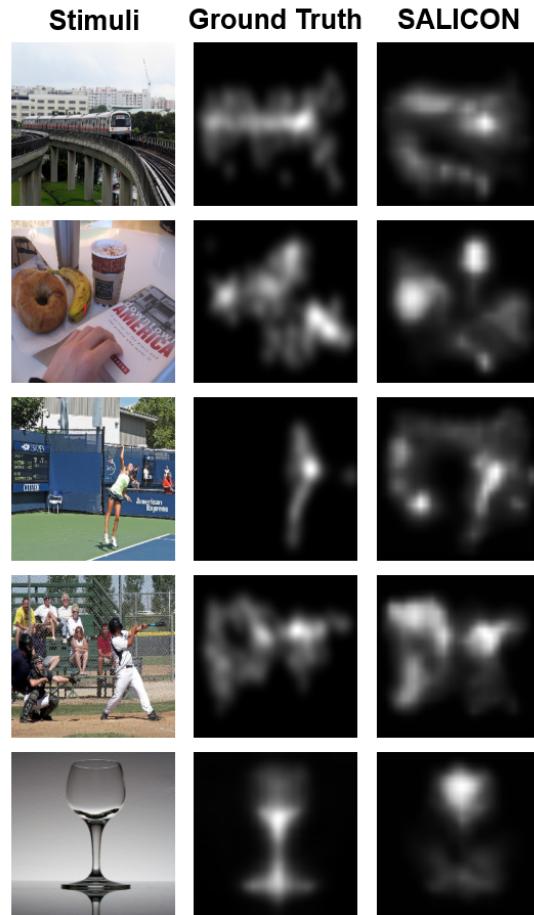
Rysunek 6: Przykładowe wyniki modeli MSI-Net

5.4 DeepGaze II

Reimplementacja modelu z pracy [Kümmerer et al., 2016]. Użyliśmy optymalizatora SGD z początkowym *learning rate* wynoszącym 0.001, oraz funkcję straty MSE. Dodatkowo zastosowany został *learning rate* malejący w zależności od numeru epoki (*inverse epoch learning rate scheduler*).

Zbiory	AUC ↑	KLD ↓	CC ↑	SIM ↑
SALICON	0.734	1.743	0.225	0.270

Tabela 5: Wyniki ilościowe modelu



Rysunek 7: Przykładowe wyniki modelu DeepGaze

5.5 Porównanie z dostępnymi modelami

Poniżej, w tabeli 6 prezentujemy zestawienie naszych modeli z dostępnymi *benchmarkami*, w tym, z oryginalnymi modelami DeepGaze oraz MSI-Net, które probowaliśmy odtworzyć.

Model	AUC ↑	KLD ↓	CC ↑	SIM ↑
DeepGaze II [Kümmerer et al., 2016]	0.873	0.424	0.770	0.664
DeepGaze IIE [Linardos et al., 2021]	0.883	0.347	0.824	0.699
MSI-Net [Kroner et al., 2020]	0.874	0.423	0.779	0.670
MLNet [Cornia et al., 2016]	0.837	0.800	0.663	0.582
Enkoder-dekoder (+ MSRA-B)	0.742	1.752	0.274	0.321
U-Net (+ DUT-OMRON)	0.699	2.215	0.228	0.300
MSI-Net (+ MSRA10K)	0.740	1.799	0.267	0.318
DeepGaze II (SALICON)	0.734	1.743	0.225	0.270

Tabela 6: Porównanie wyników modeli. W górnej części modele ogólnie dostępne. W dolnej - nasze modele i próba zaimplementowania DeepGaze II i MSI-Net. Dane zaczerpnięte z pracy [Ding et al., 2022].

6 Dyskusja

6.1 Omówienie otrzymanych rezultatów

Pomimo zadowalająco prezentujących się wyników graficznych, dane liczbowe nie odzwierciedlają pożądanej skuteczności modeli w zadaniu predykcji skupienia wzroku. Ta rozbieżność może być spowodowana przez kilka czynników:

- **Ograniczenia czasowe i technologiczne**

Brak dostępu do odpowiednio szybkiego GPU oraz ograniczenia czasowe zdecydowanie wpłynęły na możliwości wytrenowania modeli na odpowiedniej liczbie zbiorów przez dośćecznie dużo epok oraz eksperymentowania z różnymi wariantami parametrów.

- **Złe parametry uczenia**

Istotnym elementem mogą być również źle dobrane parametry trenowania, takie jak funkcje straty (*loss*) czy optymalizatory (*optimizers*). Dobre dostosowanie tych parametrów jest kluczowe dla skuteczności uczenia, a ich niedoprecyzowanie może prowadzić do sub-optimalnych wyników liczbowych. Wiąże się to z poprzednim punktem - aby móc znaleźć odpowiednią kombinację parametrów, konieczne byłoby wielokrotne wytrenowanie modeli, co w oczywisty sposób jest dużym obciążeniem czasowym i obliczeniowym.

Podsumowując, aby poprawić wyniki liczbowe, konieczne byłoby wykorzystanie większych zbiorów danych, poświęcenie znacznie większych ilości czasu na dobranie odpowiednich parametrów oraz, jeśli to możliwe, uzyskanie dostępu do bardziej zaawansowanego sprzętu z większymi zasobami obliczeniowymi.

6.2 Napotkane problemy

Podczas realizacji projektu napotkaliśmy szereg wyzwań, które miały istotny wpływ na jego przebieg oraz wyniki. Poniżej przedstawiamy główne problemy, z którymi się zetknęliśmy:

- **Reimplementacja na nowe wersje bibliotek**

Z racji szybkiego rozwoju pola uczenia maszynowego, w tym sieci neuronowych, a co za tym idzie, bibliotek z nim związanych, kody źródłowe dostępne w repozytoriach wymagały dostosowania do ich najnowszych wersji. Tyczy się to również prac, które zostały opublikowane stosunkowo niedawno (nawet implementacje sprzed 2-3 lat temu okazały się częściowo niekompatybilne). Wiązało się to z dodatkowym nakładem pracy. W kilku przypadkach nie udało nam się zaimplementować niektórych istotnych części modeli, co w rezultacie uniemożliwiło dalszą pracę z nimi.

- **Przejście z PyTorch na TensorFlow**

Niektóre modele zostały zaimplementowane w bibliotece PyTorch, z którą nie mamy żadnego doświadczenia, więc reimplementacja tych samych modeli w bibliotece TensorFlow była problematyczna. W związku z ograniczeniami czasowymi zmuszeni byliśmy porzucić plany dotyczące kilku takich modeli.

- **Niezrozumiałы opis w dostępnych pracach**

Analizując istniejące prace naukowe, napotkaliśmy na wyzwania związane z nieprecyzyjnymi opisami dotyczącymi szczegółów implementacji oraz uczenia. Wiele rzeczy było przyjętych w domyśle, bez podania ich *explicite* - byliśmy przez to zmuszeni testować wiele konfiguracji, zarówno architektur modeli jak i parametrów, i nie jesteśmy przekonani, czy udało nam się finalnie znaleźć te, które autorzy mieli na myśli.

- **Słaba jakość kodu w repozytoriach**

Niektóre repozytoria z kodem źródłowym modeli były trudne w reimplementacji z uwagi na bloki zakomentowanego kodu (co utrudniało zrozumienie jego działania), drobne niezgodności z modelem przedstawionym w pracy oraz nawet niedziałający kod, głównie z uwagi na niekompatybilność wersji odpowiednich bibliotek. Ponownie przyczyniło się to do dodatkowego nakładu pracy.

- **Drobne, lecz brzemienne w skutkach błędy w implementacji :)**

W trakcie prac napotkaliśmy na drobne błędy w naszej własnej implementacji, które na dłuższy czas pozostawały niezauważone. Chociaż dotyczyły one głównie wstępnej obróbki danych, to niedociągnięcia te skutkowały błędymi wynikami ewaluacji modeli. Poprzez utratę znacznych ilości czasu miało to wpływ na ostateczne rezultaty projektu.

Wszystkie te problemy wpływały na czasochłonność i skomplikowanie projektu, lecz wnioski z nich płynące stanowią cenną lekcję oraz wskazówki na przyszłość, umożliwiając bardziej efektywne prowadzenie podobnych projektów.

7 Konkluzje

W trakcie realizacji projektu badawczego, nasze doświadczenia przyniosły zarówno sukcesy, jak i porażki, co stanowi wartościową lekcję i przygotowanie do przyszłych przedsięwzięć naukowych.

Projekt umożliwił nam dogłębne zanurzenie się w dziedzinie predykcji skupienia wzroku przy wykorzystaniu sztucznej inteligencji. Zdobycie wiedzy praktycznej i teoretycznej w tym obszarze stanowi solidną podstawę dla naszych dalszych prac badawczych i inżynierskich.

Analiza prac naukowych stanowiła kluczowy element projektu, pozwalając nam na zrozumienie aktualnych kierunków badawczych, narzędzi i metodologii. Sam fakt zaznajomienia się z formatem i językiem prac naukowych również stanowi cenną lekcję z tego projektu.

Projekt pozwolił nam na napisanie i przetestowanie własnych implementacji modeli, co stanowi cenny wkład w naszą wiedzę praktyczną. Niemniej jednak, pomimo wysiłków, nie udało nam się w pełni odtworzyć rezultatów z prac naukowych, ani zaproponować własnego innowacyjnego rozwiązania problemu, czego przyczyny zostały omówione powyżej.

Podsumowując, projekt dostarczył nam nie tylko wiedzy merytorycznej, ale również cennych umiejętności praktycznych i doświadczeń, które z pewnością pomogą nam na naszej ścieżce akademickiej oraz w przyszłości w pracy zawodowej. Wyzwania, jakie napotkaliśmy, stanowią istotne punkty odniesienia, pomagając nam w identyfikacji obszarów, które wymagają dalszego rozwoju i doskonalenia naszych umiejętności.

Literatura

[Buswell, 1935] Buswell, G. (1935). *How People Look at Pictures: A Study of the Psychology of Perception in Art*. University of Chicago Press.

- [Cornia et al., 2016] Cornia, M., Baraldi, L., Serra, G., and Cucchiara, R. (2016). A Deep Multi-Level Network for Saliency Prediction. In *International Conference on Pattern Recognition (ICPR)*.
- [Ding et al., 2022] Ding, G., İmamoğlu, N., Caglayan, A., Murakawa, M., and Nakamura, R. (2022). SalFBNet: Learning pseudo-saliency distribution via feedback convolutional networks. *Image and Vision Computing*, page 104395.
- [Hachaj et al., 2021] Hachaj, T., Stolińska, A., Andrzejewska, M., and Czerski, P. (2021). Deep Convolutional Symmetric Encoder-Decoder Neural Networks to Predict Students' Visual Attention. *Symmetry*, 13(12).
- [Jiang et al., 2015] Jiang, M., Huang, S., Duan, J., and Zhao, Q. (2015). SALICON: Saliency in Context. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [Judd et al., 2009] Judd, T., Ehinger, K., Durand, F., and Torralba, A. (2009). Learning to predict where humans look. In *2009 IEEE 12th International Conference on Computer Vision*, pages 2106–2113.
- [Kienzle et al., 2009] Kienzle, W., Franz, M. O., Schölkopf, B., and Wichmann, F. A. (2009). Center-surround patterns emerge as optimal predictors for human saccade targets. *Journal of Vision*, 9(5):7–7.
- [Kroner et al., 2020] Kroner, A., Senden, M., Driessens, K., and Goebel, R. (2020). Contextual encoder-decoder network for visual saliency prediction. *Neural Networks*, 129:261–270.
- [Kümmerer et al., 2018] Kümmerer, M., Bylinskii, Z., Judd, T., Borji, A., Itti, L., Durand, F., Oliva, A., Torralba, A., and Bethge, M. (2018). MIT/Tübingen Saliency Benchmark. <https://saliency.tuebingen.ai/>.
- [Kümmerer et al., 2016] Kümmerer, M., Wallis, T. S., and Bethge, M. (2016). DeepGaze II: Reading fixations from deep features trained on object recognition. *arXiv preprint arXiv:1610.01563*.
- [Kummerer et al., 2018] Kummerer, M., Wallis, T. S., and Bethge, M. (2018). Saliency benchmarking made easy: Separating models, maps and metrics. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 770–787.
- [Kümmerer et al., 2022] Kümmerer, M., Bethge, M., and Wallis, T. S. A. (2022). DeepGaze III: Modeling free-viewing human scanpaths with deep learning. *Journal of Vision*, 22(5):7–7.
- [Linardos et al., 2021] Linardos, A., Kümmerer, M., Press, O., and Bethge, M. (2021). DeepGaze IIE: Calibrated prediction in and out-of-domain for state-of-the-art saliency modeling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12919–12928.
- [Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5–9, 2015, Proceedings, Part III 18*, pages 234–241. Springer.
- [Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- [Tatler, 2007] Tatler, B. W. (2007). The central fixation bias in scene viewing: Selecting an optimal viewing position independently of motor biases and image feature distributions. *Journal of Vision*, 7(14):4–4.
- [Xu et al., 2014] Xu, J., Jiang, M., Wang, S., Kankanhalli, M. S., and Zhao, Q. (2014). Predicting human gaze beyond pixels. *Journal of Vision*, 14(1):28–28.
- [Yang et al., 2013] Yang, C., Zhang, L., Lu, H., Ruan, X., and Yang, M.-H. (2013). Saliency Detection via Graph-Based Manifold Ranking. pages 3166–3173.