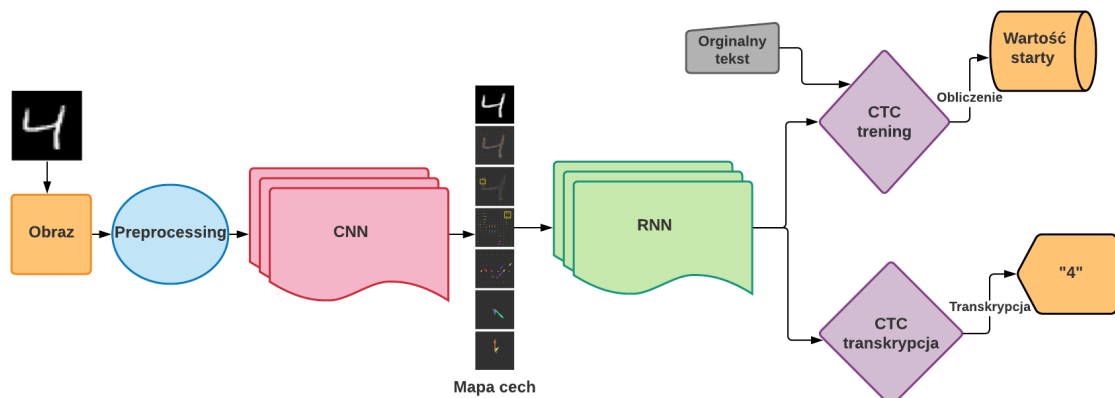


0.1 Wstęp

Przez wiele lat systemy rozpoznawania pisma odręcznego HTR stosowały modele oparte na ukrytych modelach Markowa. Jednak jak pokazują badania i prace publikowane na konferencjach ICFHR - The International Conference on Frontiers of Handwriting Recognition modele oparte na HMM są systematycznie zastępowane przez modele ANN ze względu na ich większą skuteczność i dokładność rozpoznania [?]. Ośrodek ICFHR uznawany jest za lidera w dziedzinie rozpoznawania pisma ręcznego. Jego cele to: gromadzenie międzynarodowych ekspertów ze środowisk akademickich i biznesowych, dzielenie się wiedzą i doświadczeniami oraz promowanie badań wspomagających rozwój we wszystkich aspektach rozpoznawania pisma ręcznego i ich zastosowań [?]. Uproszczony schemat działania przykładowego modelu HTR przedstawiono na rysunku 1. Podstawowe zasady działania opisanych modeli HTR można podzielić na cztery kluczowe etapy:

1. Obraz wejściowy przechodzi przez proces prepossessingu.
2. Przetworzony obraz wprowadzany jest do warstw sieci CNN w celu wyodrębnienia cech w formie mapy cech.
3. Sieć RNN propaguje odpowiednie informacje poprzez swoje warstwy zapewniając dokładniejszą charakterystykę treningu i rozpoznania.
4. Etap ten zawiera przypadek trenowania lub transkrypcji modelu HTR. Podczas trenowania system CTC - Connectionist Temporal Classification otrzymuje macierz wyjściową sieci RNN oraz oryginalny tekst obrazu zapisany w formie cyfrowej i oblicza wartość straty. Natomiast gdy model dokonuje transkrypcji obrazu system CTC otrzymuje tylko macierz i dekoduje ją na język maszynowy.



Rysunek 1: Uproszczony schemat działania przykładowego modelu HTR.

Motywacją do przeprowadzenia poniższych eksperymentów było sprawdzenie skuteczności i porównanie systemów HTR opartych na modelach sztucznych sieci neuronowych przy wykorzystaniu trzech opisanych w późniejszej części pracy architektur splotu sieci CNN oraz RNN, czyli sieci konwolucyjno-rekurencyjnej (CRNN - Convolutional Recurrent Neural Network). W celu osiągnięcia jak najdokładniejszych wyników sprawdzających jedynie skuteczność sieci CRNN użyto uniwersalnych metod preprocesingu, treningu i walidacji na tych samych trzech zbiorach dla każdej architektury.

0.2 Przygotowanie środowiska i danych

Pierwszym krokiem w przygotowaniu środowiska było zebranie i uporządkowanie użytych zbiorów pobranych z internetowej bazy. Pozostawiono oryginalne metodologie użyte do partycjonowania omawianych zbiorów na zbiory uczące, walidacyjne oraz testowe. Do eksperymentów wykorzystano jedynie obrazy zawierające tekst na poziomie linii więc obrazy przedstawiające tekst na poziomie zdań lub stron odrzucono. Obrobione zbiory danych przekształcono do jednego pliku w formacie HDF5, aby ułatwić wprowadzanie danych do modelu, zmniejszyć rozmiar oraz umożliwić szybsze ładowanie danych. Następnie utworzono nowy projekt w wirtualnym środowisku Google Colab, pobrano odpowiednie biblioteki i umieszczono w nim przetworzone zbiory. Wszystkie operacje zostały wykonane na środowisku, którego specyfikację techniczną przedstawia tabela 1.

Tabela 1: Specyfikacja techniczna użytego środowiska

Specyfikacja techniczna środowiska Google Colab	
GPU	1xTesla K80 , compute 3.7, having 2496 CUDA cores , 12GB GDDR5
CPU	1xsingle core hyper threaded Xeon Processors @2.3Ghz
RAM	~12.6 GB Available

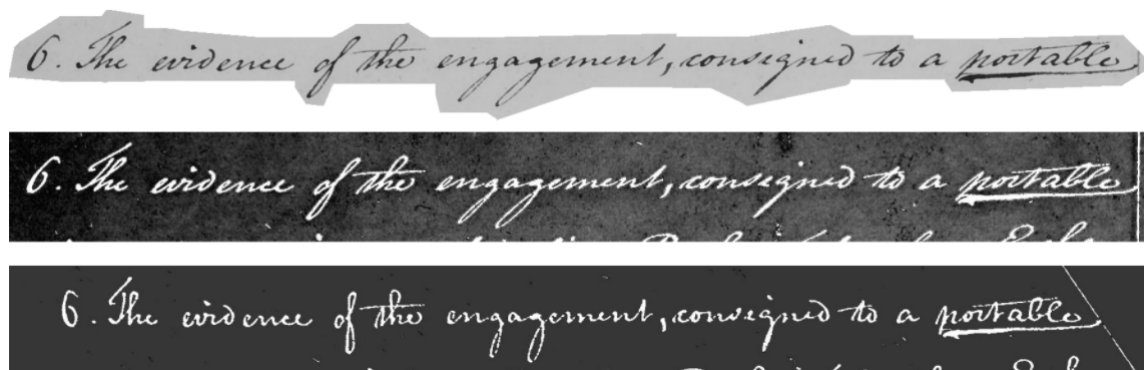
0.3 Preprocessing - Przetworzenie wstępne

Pomimo że omawiany system HTR jest w stanie przetworzyć nieobrobiony obraz oraz jest gotów do trenowania i walidacji obrazów bez wcześniejszego przetworzenia zdecydowano się zastosować metody preprocessingu. Mające na celu poprawienie wyników rozpoznania pisma odręcznego oraz zwiększenie dokładności systemu. By przetworzyć i znormalizować analizowane obrazy zastosowano szereg przekształceń między innymi: obcinanie, rotacja, binaryzowanie, skalowanie do rozmiaru 1024×128 oraz potęgowanie kontrastu obrazu. Dodatkowo zastosowano dwie zaawansowane techniki preprocessingu. Pierwszą z nich jest wydajna kompensacja oświetlenia (efficient illumination compensation) mającą na celu usunięcie cieni i zrównoważenie jasności oraz kontrastu obrazu. Technika ta została użyta i zaimplementowana z gotowego darmowego repozytorium¹. Drugą metodą jest wyrównanie kursywy (deslanting). Usunięcie kursywy to proces przekształcania obrazu odręcznego tekstu w celu zminimalizowania jego kąta nachylenia. Technika ta także została użyta i zaimplementowana z gotowego darmowego repozytorium².

Porównanie przekształceń omówionymi metodami preprocessingowymi obrazu pokazano na rysunku 2. Pierwsza linia jest losowym nieprzekształconym obrazem ze zbioru Bentham. Druga linia przedstawia ten sam obraz po przekształceniach bez użycia metod kompensacji oświetlenia oraz wyrównania kursywy. Zaś trzecia linia przedstawia obraz po wszystkich omówionych przekształceniach. Dzięki zastosowaniu przekształceń jesteśmy w stanie usunąć niepożądane szумы w obrazie dzięki czemu model HTR wytrenuje dokładniejsze wzorce i a co za tym idzie polepszy się transkrypcja analizowanego obrazu.

¹Kompensacja oświetlenia: <https://github.com/fanyirobin/text-image-binarization>

²Wyrównanie kursywy: <https://github.com/githubharald/DeslantImg>



Rysunek 2: Wizualizacja metod preprocessingowych.

0.4 Opis i implementacja architektur użytych w systemie HTR

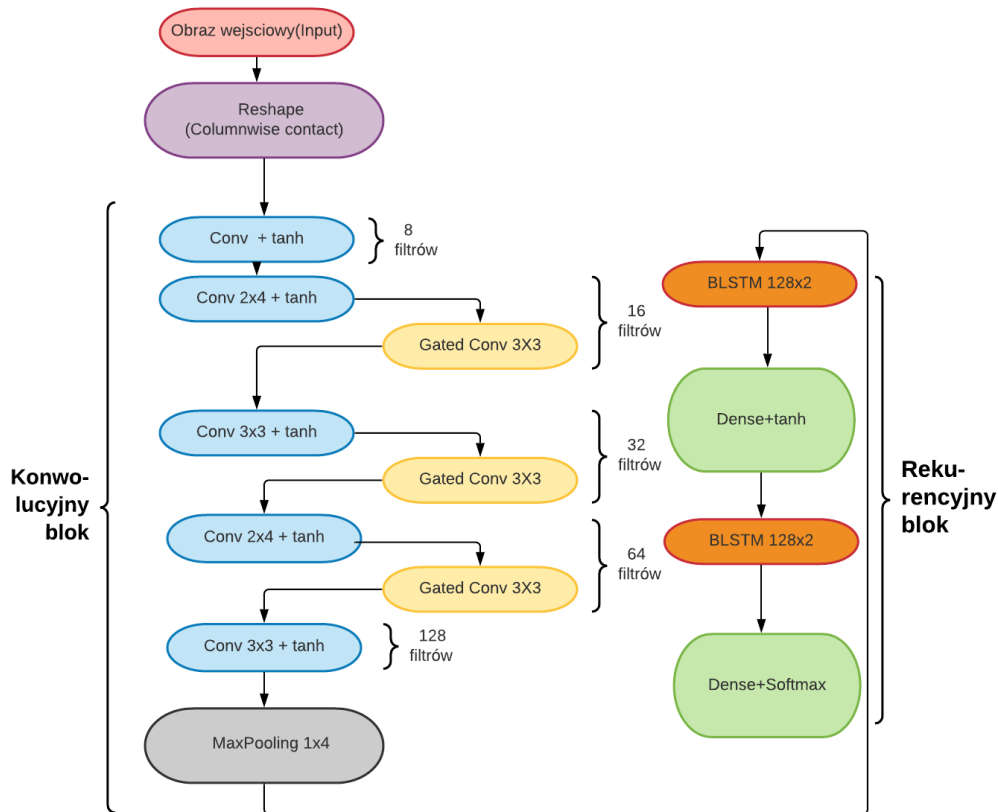
Jako inspirację dla tej części pracy posłużyły trzy modelowe podejścia z najnowocześniejszych rozwiązań problemu rozpoznawania pisma odręcznego korzystających z sieci CRNN, czyli połączenia sieci CNN i RNN. Specyfikacje oraz implementacje architektur Puigcerver, Bluche oraz Flor przedstawiono w poniższych podpunktach. Nazwy własne wszystkich przedstawionych architektur pochodzą od nazwisk autorów prac, w których zostały opisane.

0.4.1 Bluche

Pierwszą zaimplementowaną architekturą jest model zaproponowany przez Theodore Bluche w artykule „Gated Convolutional Recurrent Neural Networks for Multilingual Handwriting Recognition” [?]. Praca skupia się na opisanie modelu złożonego z sieci CNN, Gated-CNN oraz BLSTM. Technika ta ma na celu wyodrębnienie odpowiednich cech obrazu przy bardzo małej liczbie parametrów (około 730,000). Optymalna struktura i mała ilość parametrów skutkują możliwością szybkiego trenowania modelu na GPU i dekodowania na CPU [?].

Na rysunku 3 przedstawiono schemat struktury architektury Bluche (Gated-CNN-BLSTM) [?]. Jak widać na schemacie architektura składa się z dwóch bloków:

- Bloku konwolcyjnego składającego się z warstw Conv lub Gated-Conv z filtrami równymi $8n$, n odpowiednie dla n -tej warstwy. Dodatkowo dodana została funkcja \tanh służąca do aktywacji i inicjacji warstwy Conv. Pod koniec użyto warstwy *MaxPooling* wyliczającą maksymalną wartość dla każdego fragmentu mapy obiektów. Zatem omawiany blok składa się z kolejnych warstw:
 - warstwa Conv 3×3 z 8 filtrami $+ \tanh$
 - warstwa Conv 2×4 z 16 filtrami $+ \tanh$
 - warstwa Gated-Conv 3×3
 - warstwa Conv 3×3 z 32 filtrami $+ \tanh$
 - warstwa Gated-Conv 3×3
 - warstwa Conv 2×4 z 64 filtrami $+ \tanh$



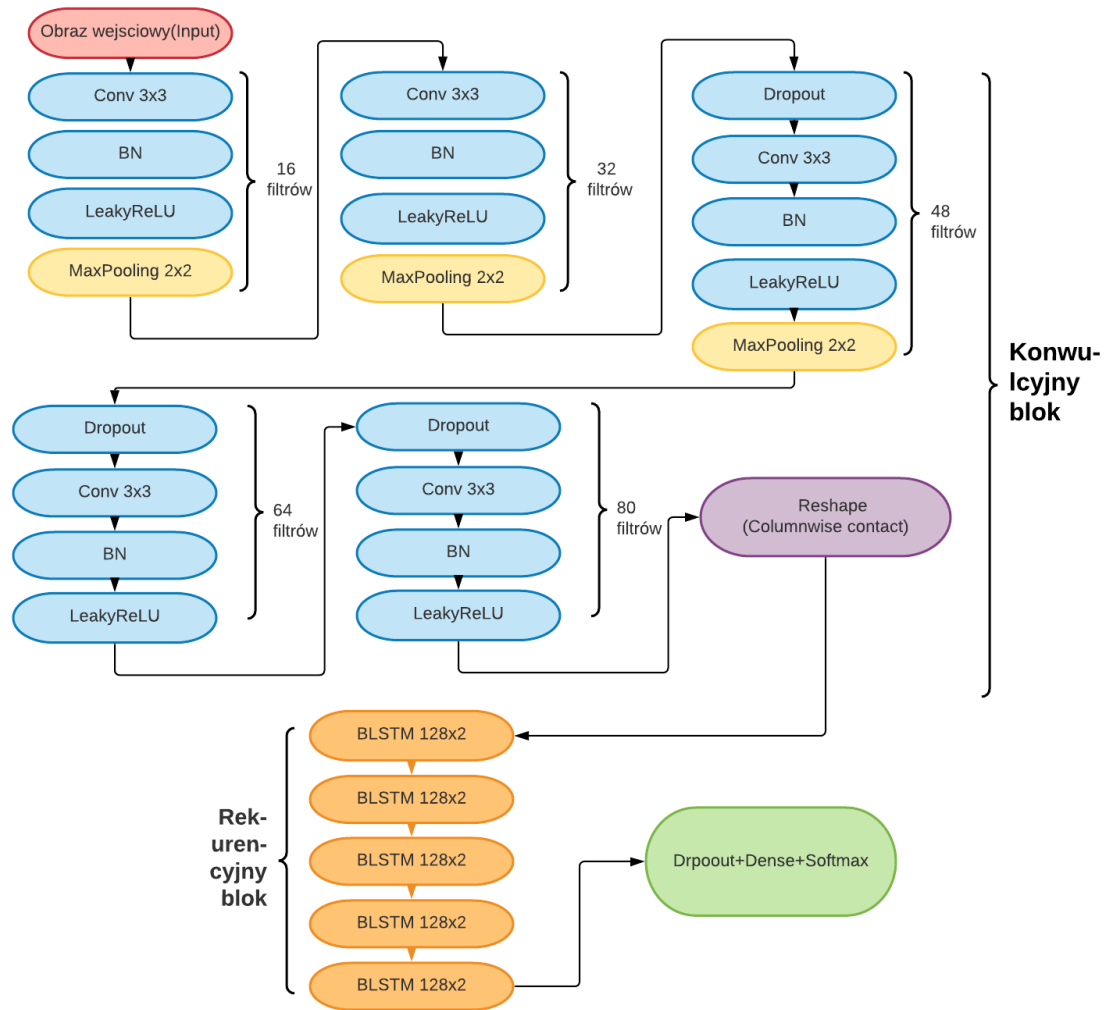
Rysunek 3: Schemat struktury architektury Bluche.

- warstwa Gated-Conv 3×3
- warstwa Conv 3×3 z 128 filtrami $+tanh$
- *Maxpooling* 1×4
- Bloku rekurencyjnego składającego się z naprzemiennych warstw BLSTM oraz całkowicie połączonych warstw *Dense* wraz z funkcjami *tanh* lub *Softmax* służącymi do aktywacji [?]. Omawiany blok składa się z kolejnych warstw:
 - warstwa BLSTM 128×2
 - *Dense + tanh*
 - warstwa BLSTM 128×2
 - *Dense*
 - *Softmax*

0.4.2 Puigcerver

Drugą zaimplementowaną architekturą jest model wprowadzony przez Joan Puigcerver w artykule „Are Multidimensional Recurrent Layers Really Necessary for Handwritten Text Recognition?” [?]. Omawiana architektura charakteryzuje się wysokim stopniem rozpoznawania i dużą liczbą parametrów (około 9,600,000).

Na rysunku 4 przedstawiono schemat struktury architektury Puigcerver (CNN-BLSTM) [?]. Jak widać na schemacie architektura składa się z dwóch bloków:



Rysunek 4: Schemat struktury architektury Puigcerver.

- Bloku konwulcyjnego składającego się z pięciu warstw Conv z filtrami równymi $16n$, n odpowiednie dla n -tej warstwy. Dodatkowo dodana została funkcja *MaxPooling* służąca do zmniejszenia danych na trzech pierwszych warstwach Conv. W celu zmniejszenia overfittingu zastosowano warstwę *Dropout*. Po warstwie Conv stosowana jest normalizacja wsadowa *BN* – *BatchNormalization* by znormalizować wejścia z funkcji nieliniowych aktywacji. Funkcja *LeakyReLU* użyta jest jako funkcja aktywacji w warstwie Conv [?]. Zatem omawiany blok składa się z kolejnych warstw:

- warstwa Conv 3×3 z 16 filtrami
- normalizacja wsadowa *BN*
- funkcja *LeakyReLU*
- *Maxpooling* 2×2
- warstwa Conv 3×3 z 32 filtrami
- normalizacja wsadowa *BN*
- funkcja *LeakyReLU*
- *Maxpooling* 2×2

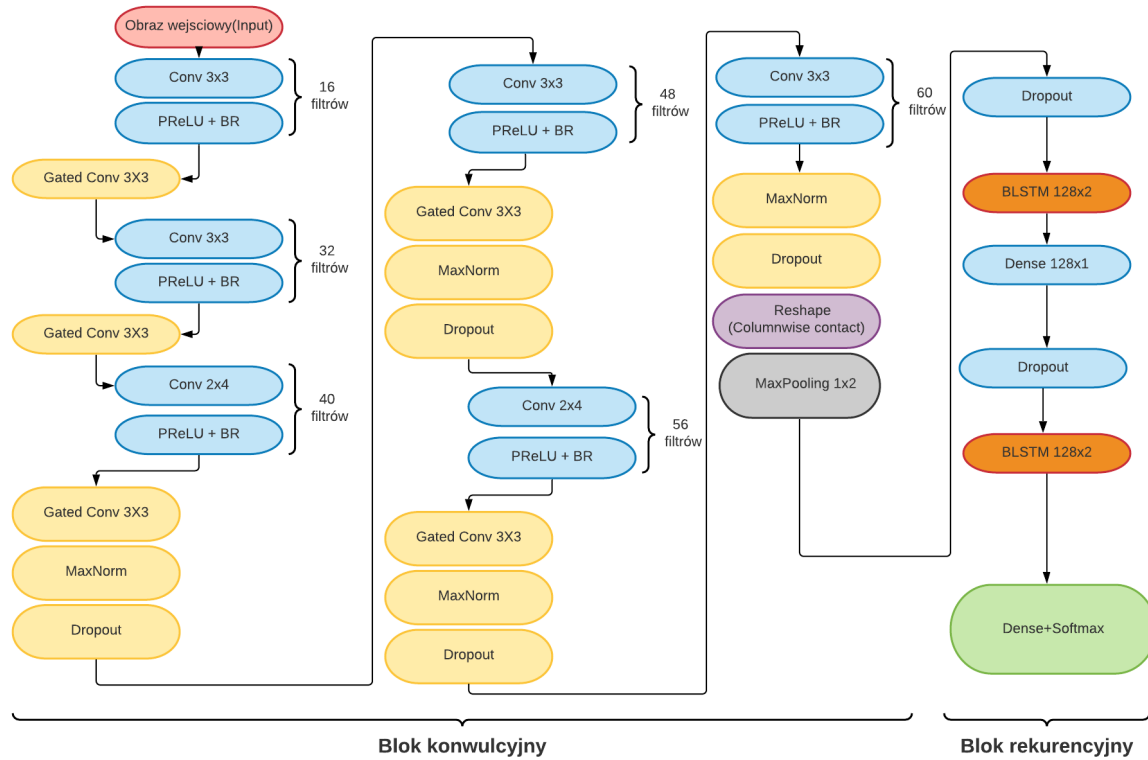
- warstwa Conv 3×3 z 48 filtrami
 - normalizacja wsadowa *BN*
 - funkcja *LeakyReLU*
 - *Maxpooling* 2×2
 - *Dropout*
 - warstwa Conv 3×3 z 64 filtrami
 - normalizacja wsadowa *BN*
 - funkcja *LeakyReLU*
 - *Dropout*
 - warstwa Conv 3×3 z 80 filtrami
 - normalizacja wsadowa *BN*
 - funkcja *LeakyReLU*
 - Reshape
- Bloku rekurencyjnego składającego się z pięciu warstw BLSTM oraz całkowicie połączonej warstwy *Dropout* wraz z *Dense* oraz *Softmax* [?]. Omawiany blok składa się z kolejnych warstw:
 - warstwa BLSTM 128×2
 - warstwa BLSTM 128×2
 - warstwa BLSTM 128×2
 - warstwa BLSTM 128×2
 - warstwa BLSTM 128×2
 - *Dropout*
 - *Dense*
 - *Softmax*

0.4.3 Flor

Trzecią zaimplementowaną architekturą jest model wprowadzony przez Arthur Flor de Sousa w artykule „Towards the Natural Language Processing as Spelling Correction for Offline Handwritten Text Recognition Systems” [?]. Omawiana architektura została zainspirowana wcześniej przedstawionymi architekturami Bluche (patrz 0.4.1) oraz Puigcerver (patrz 0.4.2) łącząc ich najlepsze cechy [?]. Ich połączenie ma na celu osiągnięcie dokładności i wyników podobnych jak Puigcerver, a zarazem utrzymanie niewielkiej liczby parametrów (około 930,000), a co za tym idzie szybkości porównywalnej do modelu Bluche. Rysunek 5 pokazuje poszczególne warstwy w blokach.

Na rysunku 5 przedstawiono schemat struktury architektury Flor (Gated-CNN-BLSTM) [?]. Jak widać na schemacie architektura składa się z dwóch bloków:

- Bloku konwulcyjnego składającego się z sześciu warstw Conv z filtrami równymi $8n$, n odpowiednie dla $(n + 1)$ -tej warstwy i pięciu Gated-Cov. Jako aktywator i



Rysunek 5: Schemat struktury architektury Flor.

inicjator używamy funkcji *PReLU*. Po warstwie Conv stosowana jest renormalizacja wsadowa *RN* – *BatchRenormalization* by znormalizować wejścia z funkcji nieliniowych aktywacji. Dodatkowo dodana została funkcja *MaxNorm* służąca do zmniejszenia danych na czterech ostatnich warstwach Conv. W celu zmniejszenia overfittingu zastosowano warstwę *Dropout*. Funkcja *LeakyReLU* użyta jest jako funkcja aktywacji w warstwie Conv [?]. Zatem omawiany blok składa się z kolejnych warstw:

- warstwa Conv 3×3 z 16 filtrami
- funkcja *PReLU*
- renormalizacja wsadowa *BR*
- warstwa Gated-Conv 3×3
- warstwa Conv 3×3 z 32 filtrami
- funkcja *PReLU*
- renormalizacja wsadowa *BR*
- warstwa Gated-Conv 3×3
- warstwa Conv 2×4 z 40 filtrami
- funkcja *PReLU*
- renormalizacja wsadowa *BR*
- warstwa Gated-Conv 3×3

- funkcja *MaxNorm*
 - *Dropout*
 - warstwa Conv 3×3 z 48 filtrami
 - funkcja *PReLU*
 - renormalizacja wsadowa *BR*
 - warstwa Gated-Conv 3×3
 - funkcja *MaxNorm*
 - *Dropout*
 - warstwa Conv 2×4 z 56 filtrami
 - funkcja *PReLU*
 - renormalizacja wsadowa *BR*
 - warstwa Gated-Conv 3×3
 - funkcja *MaxNorm*
 - *Dropout*
 - warstwa Conv 3×3 z 60 filtrami
 - funkcja *PReLU*
 - renormalizacja wsadowa *BR*
 - funkcja *MaxNorm*
 - *Dropout*
 - Reshape
- Bloku rekurencyjnego składającego się z dwóch warstw BLSTM oraz całkowicie połączonej warstwy *Dropout* wraz z *Dense* oraz *Softmax* [?]. Omawiany blok składa się z kolejnych warstw:
 - *Dropout*
 - warstwa BLSTM 128×2
 - *Dense*
 - *Dropout*
 - warstwa BLSTM 128×2
 - *Dense*
 - *Softmax*

0.4.4 Porównanie i podsumowanie architektur

Wszystkie opisane architektury złożone są z sieci CRNN składających się z bloków konwulcyjnego i rekurencyjnego. Architektura Bluche (Gated-CNN-BLSTM) skupia się na wykorzystaniu warstw Gated-CNN, aby wyodrębnić bardziej istotne cechy obrazu (feature map). Natomiast architektura Puigcerver (CNN-BLSTM) skupia się na wykorzystaniu warstw BLSTM do rozpoznania tekstu.

Tabela 2: Statystyki parametrów dla zaproponowanych architektur

ilość \ model	Bluche	Flor	Puigcerver
Wszystkie parametry	729194	922354	9591202
Parametry trenowalne	729194	921074	9590722
Parametry nietrenowalne	0	1280	480

Tabela 2 przedstawia podsumowanie liczby parametrów wykorzystywanych w zaproponowanych architekturach. Można z niej wyczytać, iż Bluche w porównaniu do podejścia Puigcerver składa się z bardzo małej liczby parametrów trenowanych oraz nie posiada parametrów nie trenowanych dzięki czemu jest bardziej kompaktowa i szybsza, ale przez co prawdopodobnie jest mniej dokładna. Natomiast architektura Flor korzystająca z warstw Gated-CNN-BLSTM zachowuje optymalną liczbę parametrów zapewniającą szybkość i dokładność.

0.4.5 Implementacja architektur

Wszystkie omówione architektury zaimplementowano korzystając z gotowych rozwiązań dostępnych w bibliotekach Tensorflow 2.0 oraz Keras. Następnie stworzono rozwiązanie, dzięki któremu użytkownik poprzez komendy odwołujące się do głównego pliku projektu może wybrać architekturę oraz zbiór danych, z których chce korzystać.

0.5 Trenowanie architektur na zbiorach

Po zaimplementowaniu opisanych w poprzednim podrozdziale architektur (patrz rozdział 0.4) skupiono się na wytrenowaniu zaproponowanych modeli na przygotowanych wcześniej zbiorach (patrz rozdział ??). W poniższych paragrafach skupiono się na opisanu procesu trenowania na każdym z zaproponowanych zbiorów. Każda architektura trenowana jest na identycznym zbiorze treningowym, walidacyjnym i testowym. Wszystkie modele zostały przeszkolone, aby zminimalizować wartość utraty walidacji (validation loss value) funkcji CTC zgodnie z teorią opisaną w poprzednich rozdziałach (patrz rozdziały ??, ??). Oryginalne obrazy zostały podane preprocessingowi i przekształcone do rozmiaru $1024 \times 128 \times 1$. Proces trenowania został podzielony na 500 epoch z batch (partia) o rozmiarze 16. Dodatkowo została zastosowana technika Early Stopping po 20 epoch bez poprawy wartości utraty walidacji (validation loss value). W celu uzyskania jak najlepszych wyników dla każdego modelu zastosowano tolerancję 10 epoch bez poprawy validation loss value, po której stosowana jest technika zmniejszenia szybkości nauki (learning rate) o wartość 0.2. Do uczenia zastosowano optymalizator RMSProp z biblioteki Keras ze standardową szybkością uczenia się każdego modelu, aby przyrostowo aktualizować parametry za pomocą gradientów strat modelu CTC. Tabela 3 przedstawia zbiorcze statystyki opisujące trening architektur na omawianych zbiorach.

Tabela 3: Zbiorcze statystyki opisujące trening architektur

Baza	Model	l.treno- wane obrazy	l. walido- wane obrazy	Batch	Śr. czas treningu na obraz	Najlepsza epoch
Bentham	Bluche	8807	1372	16	0.005001[s]	91
	Flor				0.009876[s]	84
	Puigcerver				0.010216[s]	101
Iam	Bluche	6161	1840	16	0.003982[s]	61
	Flor				0.009289[s]	137
	Puigcerver				0.006625[s]	128
Saintgall	Bluche	468	235	16	0.004403[s]	70
	Flor				0.013447[s]	61
	Puigcerver				0.007975[s]	58

Z tabeli 2 można wywnioskować, iż najszybszym modelem w kontekście treningu obrazu jest model oparty na architekturze Bluche. Najwolniejszym modelem przy transkrypcji próbki ze zbioru Iam jest model Flor natomiast dla zbiorów Bentham i Saintgall model Puigcerver.

0.5.1 Bentham

Baza danych Bentham została przekształcona i wytrenowano na niej wszystkie architektury wyżej wymienionymi metodami. Na rysunku 6 przedstawiono krzywe straty treningowej i walidacyjnej dla wszystkich omawianych architektur na zbiorze Bentham. Rysunek 6a przedstawia porównanie straty treningowej i walidacyjnej dla każdej architektury z osobna, zaś rysunek 6b przedstawia łączne zestawienie krzywych strat treningowych i walidacyjnych na zbiorze Bentham.

Z wykresów 6 oraz tabeli 3 wynika, iż najmniejszą liczbę epoch do zatrzymania trenowania potrzebowała architektura Flor, a największa Puigcerver. Straty treningowe i walidacyjne dla początkowych epoch w Flor są dużo większe niż dla pozostałych architektur. Mimo tego statystyka ukazująca czas treningu na jeden obraz pokazuje, iż najszybszą architekturą jest Bluche, a najwolniejszą Puigcerver.

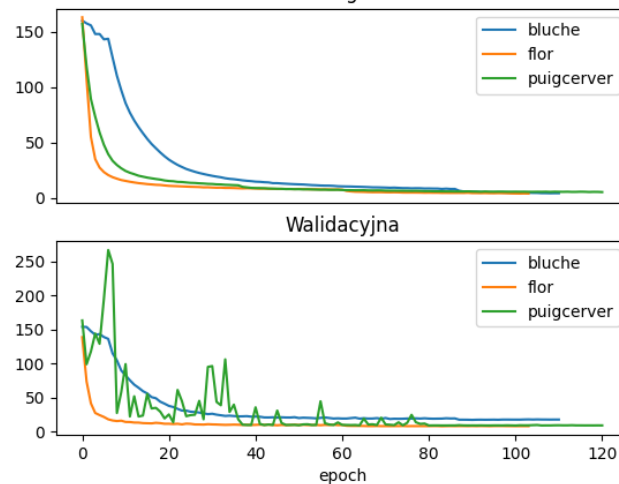
0.5.2 Iam

Baza danych Iam została przekształcona i wytrenowano na niej wszystkie architektury wyżej wymienionymi metodami. Na rysunku 7 przedstawiono krzywe straty treningowej i walidacyjnej dla wszystkich omawianych architektur na zbiorze Iam. Rysunek 7a przedstawia porównanie straty treningowej i walidacyjnej dla każdej architektury, z osobna zaś rysunek 7b przedstawia łączne zestawienie krzywych strat treningowych i walidacyjnych na zbiorze Iam.

Z wykresów 7 oraz tabeli 3 wynika, iż najmniejszą liczbę epoch do zatrzymania trenowania potrzebowała architektura Bluche, a największa Flor. Straty treningowe i walidacyjne dla początkowych epoch są największe w Bluche, natomiast dla Puigcerver najmniejsze. Statystyka ukazująca czas treningu na jeden obraz pokazuje, iż najszybszą architekturą jest Bluche, a najwolniejszą Flor.



(a) Krzywe straty treningowej i walidacyjnej.

Zbiorcze krzywe straty treningowej i walidacyjnej zbiorze bentham

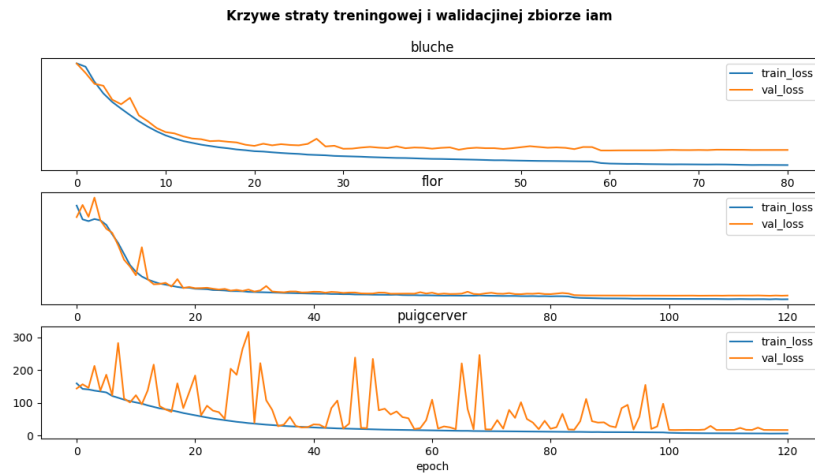
(b) Zbiorcze krzywe straty treningowej i walidacyjnej.

Rysunek 6: Krzywe straty treningowej i walidacyjnej architektur na zbiorze Bentham.

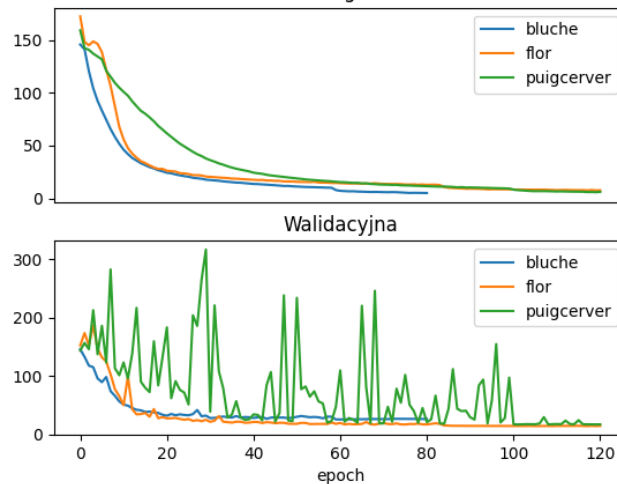
0.5.3 Saintgall

Baza danych Saintgall została przekształcona i wytrenowano na niej wszystkie architektury wyżej wymienionymi metodami. Na rysunku 8 przedstawiono krzywe straty treningowej i walidacyjnej dla wszystkich omawianych architektur na zbiorze Saintgall. Rysunek 8a przedstawia porównanie straty treningowej i walidacyjnej dla każdej architektury z osobna, zaś rysunek 8b przedstawia łączne zestawienie krzywych strat treningowych i walidacyjnych na zbiorze Saintgall.

Z wykresów 8 oraz tabeli 3 wynika, iż najmniejszą liczbę epoch do zatrzymania trenowania potrzebowała architektura Flor, a największa Bluche. Straty treningowe i walidacyjne dla początkowych epoch w Flor są dużo większe niż dla pozostałych architektur. Mimo tego statystyka ukazująca czas treningu na jeden obraz pokazuje, iż najszybszą architekturą jest Bluche, a najwolniejszą Flor.



(a) Krzywe straty treningowej i walidacyjnej.

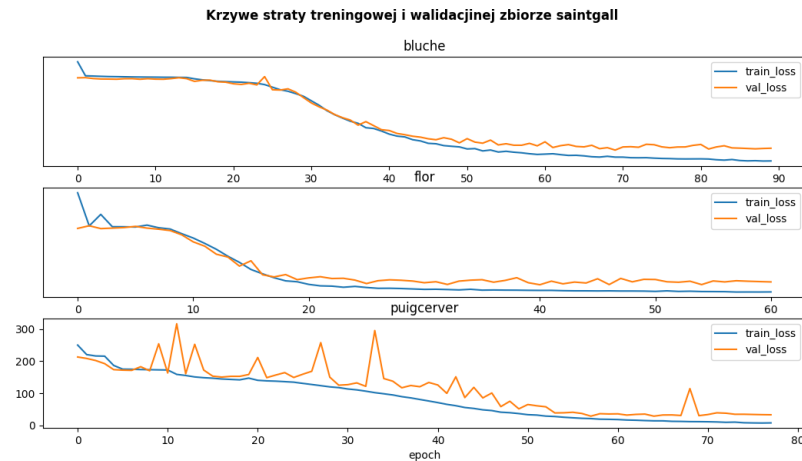
Zbiorcze krzywe straty treningowej i walidacyjnej zbiorze iam

(b) Zbiorcze krzywe straty treningowej i walidacyjnej.

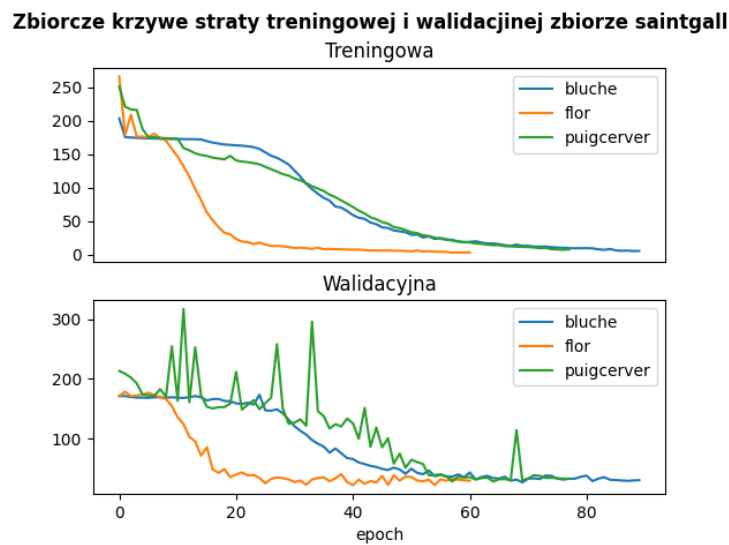
Rysunek 7: Krzywe straty treningowej i walidacyjnej architektury na zbiorze Iam.

0.5.4 Podsumowanie procesu trenowania

Dla wszystkich przedstawionych zbiorów zauważyć można skokowość krzywej straty wartości walidacyjnej dla architektury Puigcerver. Zależność ta najprawdopodobniej wynika z struktury sieci BLSTM. Analiza krzywych uczenia się architektury Flor oraz Puigcerver wykazuje dobre dopasowanie, natomiast zauważamy, że utrata treningu i utrata walidacji nie są ze sobą skorelowane w architekturze Bluche oraz Flor trenowanym na zbiorze Saint-gall. Oznacza to, że w miarę zmniejszania się utraty szkolenia, utrata walidacji pozostaje taka sama. Zatem model nie jest ulepszany i nadmiernie dopasowuje się do danych uczących (lekki overfitting). Problem ten można rozwiązać poprzez zmniejszenie parametrów szybkości uczenia lub techniki Early Stopping. W tym przypadku zdecydowano się jednak nie dopasowywać tych parametrów i przyjęto zaprezentowane wytrenowanie modelu, by nie wpływać na porównawcze wyniki wszystkich architektur. Biorąc pod uwagę parametr średniego czasu treningu na jeden obraz można jednoznacznie stwierdzić, że architektura Bluche jest najszybsza (mała liczba parametrów), jednak nie można stwierdzić który model



(a) Krzywe straty treningowej i walidacyjnej.



(b) Zbiorcze krzywe straty treningowej i walidacyjnej.

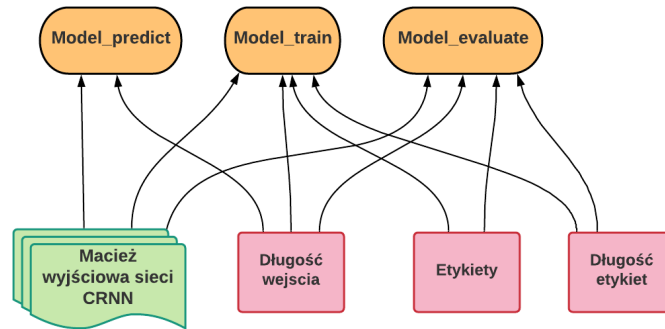
Rysunek 8: Krzywe straty treningowej i walidacyjnej architektur na zbiorze Saintgall.

jest najwolniejszy.

0.6 Model CTC

Użyty model CTC może być postrzegany jako rozszerzenie modelu biblioteki Keras przekształconego do wykonywania kroków szkolenia i dekodowania modelu HTR. Model CTC potrzebuje macierzy wejścia i wyjścia sieci CRNN oraz przy etapie trenowania etykiety zawierającej oryginalny tekst. CTC składa się z 3 modeli z biblioteki Keras: model uczenia (model_train) posiadający 4 wejścia, model przewidywania (model_predict) posiadający 2 wejścia oraz model ewaluacji wyników (model_evaluate) posiadający 4 wejścia. Schemat modelu CTC jest przedstawiony na rysunku 9. Modele te są automatycznie definiowane i zapisywane podczas kompilacji modelu CTC. Każdy z opisanych modeli ma określoną funkcję straty (związaną z konkretnym zadaniem).

Model_predict do dekodowania tekstu używa znaków z tabeli ASCII zawierających 95 unikatowych rekordów.



Rysunek 9: Schemat modelu CTC.

W części projektowej została użyta i zaimplementowana paczka z biblioteki Tensflow 2.0 autorstwa Harald Scheidl napisana w języku programowania C++ zawierająca model CTC Word Beam Search³ [?]. Model ten użyto dla wszystkich modeli HTR opartych na opisanych wyżej architekturach.

0.7 Wyniki

Miary oceny błędu stosowane do walidacji wyników zostały obliczone z metodą odległości Levenshteina pomiędzy oryginalnym a przewidywanym tekstem. Metryki zostały dokładnie opisane w rozdziale ???. Do oceny walidacji wyników użyto miar: CER - współczynnik błędów znaków, WER - współczynnik błędów słów, SER - współczynnik błędów zdań. W poniższych paragrafach skupiono się na zwizualizowaniu i opisaniu wyników miar oceny walidacji oraz szybkości walidacji zaproponowanych modeli.

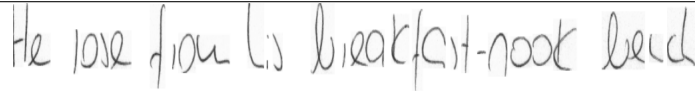
0.7.1 Wizualizacja predykcji wybranych obrazów

W celu ukazania zasady działania i wizualizacji wyników modelu HTR stworzono zestawienie przedstawiające jedną losowo wybraną próbę ze zbioru testowego wraz z predykcjami obliczonymi przez model HTR. Następnie dla wybranej próbki obliczono miary oceny błędu pomiędzy predykcją a oryginalnym tekstem. Opisany eksperyment powtórzono dla próbek ze wszystkich zbiorów danych dla kolejnych modeli HTR korzystających z architektur Bluche, Flor oraz Puigcerver. Otrzymane wyniki umieszczono w tabelach (4, 5, 6). Pierwszy wiersz w każdej tabeli przedstawia analizowany obraz zawierający oryginalny tekst zapisany pismem odręcznym. Kolejne wiersze zawierają kolejno oryginalny tekst w języku maszynowym oraz transkrypcje wykonane przez modele HTR korzystające z odpowiedniej architektury. Kolorem czerwonym podkreślono błędy transkrypcji modelu HTR a oryginalnego tekstu na poziomie znaku.

W tabeli 4 przedstawiono wizualizację predykcji modeli opartych na różnych architekturach dla próbki ze zbioru Iam. Zauważono, że najlepiej z transkrypcją na poziomie znaku poradził sobie model oparty na architekturze Flor a na poziomie słowa model Bluche. Każdy z modeli HTR dokonał błędów w transkrypcji zdania zatem wartość miary SER jest równa 1.

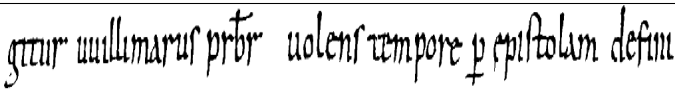
³Link do użytej paczki zawierającej model CTC Word Beam Search: <https://github.com/githubharald/CTCWordBeamSearch>

Tabela 4: Wizualizacja predykcji dla próbki ze zbioru Iam

Obraz z bazy Iam					
Orginalny tekst	He rose from his breakfast-nook bench		CER	WER	SER
Bluche	He rose from his Bireakfastnoot ben		0.162	0.5	1
Flor	He lose from Lis breakfart-nook bech		0.135	0.833	1
Puigcerver	He rose from his breakfart-nook bend		0.081	0.333	1

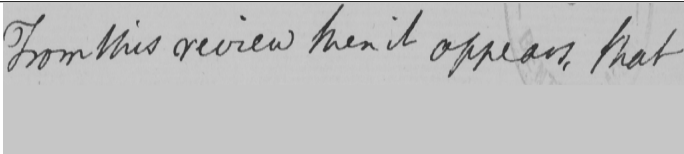
Kolejno w tabeli 5 przedstawiono wizualizację predykcji modeli opartych różnych architekturach dla próbki ze zbioru Saintgall. Oryginalny obraz jest bardzo słabo czytelny i ciężko jest odczytać oryginalny tekst na podstawie samego manuskryptu. Mimo tego zaproponowane modele HTR radzą sobie bardzo dobrze w transkrypcji. Niskie wyniki miar ocen błędu wynikają z struktury oraz sposobu zapisu tekstu wykonanego przez tylko jednego autora dzięki czemu modele były w stanie wyodrębnić pożądane cechy.

Tabela 5: Wizualizacja predykcji dla próbki ze zbioru Saintgall

Obraz Saintgall					
Orginalny tekst	gitur willmarus prbr volens tempore p epistolam defini		CER	WER	SER
Bluche	gitur vuillmarus prbr volens tempore p epislolam defini		0.054	0.25	1
Flor	gitur villmarus prbr volens tempore pepistolam defini		0.036	0.375	1
Puigcerver	gitur villmarus prbr volens tempore i epistolam defui		0.091	0.5	1

Natomiast w tabeli 6 przedstawiono wizualizację predykcji modeli opartych na różnych architekturach dla próbki ze zbioru Bentham. Z tabeli 6 można wywnioskować, że najlepszą transkrypcję na poziomie znaku oraz słowa otrzymuje model oparty na architekturze Flor. Zauważono iż żaden z modeli poprawnie nie wykrył znaku interpunkcyjnego „ , ” (przecinka). Znak ten jest pomijany lub błędnie interpretowany. Przy analizie większej ilości próbek testowych zauważono, że architektury mają bardzo duży problem z poprawnym rozpoznaniem znaków interpunkcyjnych oraz akcentu. W dalszej części pracy (rozdział 0.7.3) przedstawiono zbiorcze wyniki miar oceny problemu dla całego zbioru testowego oraz przedstawiono skale problemu rozpoznawania znaków specjalnych.

Tabela 6: Wizualizacja predykcji dla próbki ze zbioru Bentham

Obraz z bazy Bentham					
Originalny tekst	From this review then it appears, that	CER	WER	SER	
Bluche	From this orevien then t appears that	0.132	0.428	1	
Flor	From this review then it appeas s that	0.079	0.25	1	
Puigcerver	From this orecrew hen it appears that	0.132	0.429	1	

0.7.2 Szybkość transkrypcji modeli

Używając zbiorów testowych zawierających te same obrazy testowe z baz Iam, Bentham, Saintgall obliczono szybkość transkrypcji przez wszystkie opisane architektury. Tabela 7 przedstawia zestawienie liczb testowanych obrazów danego zbioru oraz średni czas potrzebny do wykonania jednego przekształcenia obrazu przez model HTR.

Tabela 7: Statystyki prędkości modeli przy transkrypcji obrazu

Baza	Model	Ilość testowanych obrazów	Średni czas przekształcenia obraz
Bentham	Bluche	820	0.082999 [s]
	Flor		0.118023 [s]
	Puigcerver		0.065746 [s]
Iam	Bluche	1861	0.060051 [s]
	Flor		0.115691 [s]
	Puigcerver		0.062677 [s]
Saintgall	Bluche	707	0.064852 [s]
	Flor		0.160572 [s]
	Puigcerver		0.073350 [s]

Z tabeli 7 można wywnioskować, iż najwolniejszym modelem w kontekście szybkości rozpoznania obrazu wbrew początkowym założeniom nie jest model oparty na architekturze Puigcerver, lecz model Flor. Co zaskakujące, najszybszym modelem przy transkrypcji próbki ze zbioru Bentham jest właśnie model Puigcerver posiadający największą ilość parametrów. Dla pozostałych baz zgodnie z oczekiwaniami najszybszy jest model Bluche posiadający najmniejszą liczbę parametrów. Należy jednak zaznaczyć, że parametr szybkości transkrypcji obrazu w modelu HTR nie jest tak ważny jak jakość i dokładność rozpoznania, gdyż w omawianym w tej pracy podejściu skupiono się nie na efektywności, lecz dokładności modeli HTR.

0.7.3 Zbiornicze miary oceny błędu

Testy skuteczności rozpoznania przeprowadzono używając odpowiednio tych samych obrazów ze zbioru testowego odpowiednio z baz Bentham, Iam, Saintgall dla wszystkich opisanych architektur. Zbiór Iam posiada największą ilość obrazów testowych: 1861, natomiast zbiory testowe Saintgall i Bentham są o połowę mniejsze (odpowiednio: 707, 820). Można zatem przyjąć, że wartości miar ocen błędu są najdokładniejsze dla zbioru Iam,

gdyż zostały obliczone na największej próbce testowej. Po przeprowadzeniu opisanych eksperymentów wyniki miary oceny błędów umieszczono w tabeli 8.

Tabela 8: Zbiorowe miary oceny błędu modeli wyliczone na zbiorze testowym

Baza	Model	CER	WER	SER
Bentham	Bluche	0.13009	0.40865	0.90976
	Flor	0.07383	0.32301	0.79512
	Puigcerver	0.07477	0.31365	0.8
Iam	Bluche	0.13371	0.37763	0.9554
	Flor	0.07761	0.25371	0.86943
	Puigcerver	0.07114	0.21674	0.82161
Saintgall	Bluche	0.11135	0.48431	0.99717
	Flor	0.07674	0.35512	0.98444
	Puigcerver	0.12254	0.49899	1

Tabela 8 ukazuje, że dla wszystkich kombinacji architektur testowanych na wszystkich zbiorach testowych otrzymujemy dobre wyniki rozpoznania. Współczynnik błędów znaków znajduje się w przedziale 7% – 13%, zatem stosując bardzo duże uproszczenie można uznać, iż "co dziesiąty znak jest błędnie rozpoznany". Wartość procentowa współczynnika błędów słów znajduje się w dość dużym w przedziale 21% – 50% a współczynnik błędów słów występujących w analizowanej linii mieści się w przedziale 79% – 100%. Wyniki te mogą wydawać się bardzo wysokie i można na ich podstawie wysnuć tezę, że zaprezentowane modele HTR źle radzą sobie z rozpoznawaniem słów oraz zdań. Trzeba jednak wziąć pod uwagę, iż by słowo bądź zdanie było uznane jako niepoprawnie rozpoznane wystarczy tylko jedno błędne rozpoznanie znaku w zdaniu. Zatem szansa, że taki błąd wystąpi jest ogromna a z tym duże są wartości miar WER i CER. Zatem zgodnie z oczekiwaniami teoretycznymi wyniki dla każdej kombinacji przedstawionej na tabeli 8 ukazują zależność $CER < WER < SER$.

Wyniki miar oceny modeli obliczonych na bazach Bentham i Iam uzyskują najlepsze wyniki przy zastosowaniu architektur Flor oraz Puigcerver. Mimo bardzo zbliżonych wyników zauważyć można, że model Flor radzi sobie lepiej z transkrypcją zbioru testowego Bentham, natomiast model Puigcerver ze zbiorem Iam. Wyniki tych modeli są zbliżone i zadowalające (miara CER na poziomie 7% – 8%, WER 20% – 30%). Natomiast model Bluche przetestowany na tych zbiorach osiągnął rozczarujący wynik miary CER większy od pozostałych modeli o prawie 5% . Dla zbioru testowego z bazy Saintgall niezaprzeczalnie najdokładniejszym modelem jest Flor z miarą CER na poziomie 7% oraz miary WER około 35%. Następny w kwestii dokładności jest model Bluche, a najmniej dokładny model Puigcerver, który nie wykonał ani jednej idealnej transkrypcji na bazie 707 obrazów ze zbioru testowego Saintgall.

W rozdziale 0.7.1 zauważono problem modeli HTR z transkrypcją obrazów zawierających znaki interpunkcyjne oraz akcent. Obliczono, że dla zbioru Bentham procent błędnego rozpoznania znaków specjalnych to około 20 – 25% natomiast dla zbioru Iam około 10 – 15%. By zaobserwować skalę problemu i jego wpływ na wyniki zbiorowych miar oceny błędu predykcji wykonano ponownie testy miar oceny błędu, nie biorąc pod uwagę opisanych wcześniej znaków specjalnych. Wyniki eksperymentu przedstawiono w tabeli 9.

Tabela 9: Zbiorowe miary oceny błędu modeli wyliczone na zbiorze testowym z pominięciem znaków specjalnych

Baza	Model	CER
Bentham	Bluche	0.12433
	Flor	0.07098
	Puigcerver	0.07221
Iam	Bluche	0.13191
	Flor	0.07569
	Puigcerver	0.06876
Saintgall	Bluche	0.11135
	Flor	0.07674
	Puigcerver	0.12254

Wyniki miar oceny CER dla zbioru Bentham oraz Iam uległy poprawie dla wszystkich architektur, natomiast dla zbioru Saintgall pozostały identyczne. Brak zmiany miary CER na zbiorze Saintgall wynika z braku znaków specjalnym w całym zbiorze testowym. Modele walidowane na zbiorze Bentham uzyskały wzrost wartości miary CER od 0.3% aż do 0.6% natomiast na zbiorze Iam o około 0.2% punktów procentowych.

0.7.4 Podsumowanie wyników

Wyniki testów szybkości modeli przy transkrypcji obrazu jasno ukazują, iż najwolniejszy jest model Flor, natomiast prawie zawsze najszybszy model Bluche, poza przypadkiem testów wykonanych na zbiorze Bentham, gdzie najszybszy był model Puigcerver. Analizując przedstawione w tabeli 8 wyniki zbiorowych miar oceny błędu dla modeli wyliczone na zbiorze testowym można uznać, iż dla obrazów ze zbioru Saintgall i Bentham najdokładniejszą architekturą jest Flor, zaś dla zbioru Iam Puigcerver. Ze wszystkich przedstawionych modeli najbardziej uniwersalny jest Flor, a najmniej Bluche. Wszystkie przedstawione modele są dokładne, a wyniki miar błędu są zadowalające (CER na poziomie 7% – 8%, WER 20% – 30%). Omawiane modele mają duży problem z rozpoznawaniem znaków specjalnych. Problem ten dotyczy tylko zbiorów Iam i Bentham i stanowi od 0.3% aż do 0.6% błędu miary CER w zależności od wybranej bazy i architektury.