Politechnika Rzeszowska im. Ignacego Łukasiewicza

Wydział Elektrotechniki i Informatyki

KATEDRA

POLITECHNIKA
RZESZOWSKA
im. IGNACEGO ŁUKASIEWICZA

PRACA INŻYNIERSKA

DAWID KOWALCZUK

PRZYGOTOWANIE PRACY DYPLOMOWEJ W SYSTEMIE LaTeX

PROMOTOR:
dr hab. inż. Maciej Kusy, prof. PRz

Rzeszów 2026

OŚWIADCZENIE AUTORA PRACY

Oświadczam, świadomy odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem osobiście i samodzielnie, i nie korzystałem ze źródeł innych niż wymienione w pracy.

.....

PODPIS

Rzeszów University of Technology

Faculty of Electrical Engineering and Computer Science

DEPARTMENT OF

POLITECHNIKA
RZESZOWSKA
im. IGNACEGO ŁUKASIEWICZA

ENGINEERING THESIS

DAWID KOWALCZUK

THESIS IN LaTeX

SUPERVISOR:
Assoc. Prof. D.Sc. Eng.
Maciej Kusy

Rzeszów 2026

Serdecznie dziękuję . . . tu ciąg dalszych podziękowań np. dla promotora, żony, sąsiada itp.

Spis treści

1. Wstęp

1.1. Wprowadzenie do problematyki

Analiza danych nieustrukturyzowanych, w szczególności obrazów, stanowi istotny obszar rozwoju współczesnych systemów informatycznych. W sektorach takich jak medycyna, przemysł czy archiwizacja cyfrowa, dane wizualne stanowią coraz większą część przetwarzanych informacji. Przez ostatnią dekadę standardem w ich analizie były metody wizji komputerowej oparte na splotowych sieciach neuronowych. Choć rozwiązania te sprawdzają się w zadaniach klasyfikacji, okazują się niewystarczające w bardziej złożonych procesach wymagających głębokiego zrozumienia semantyki sceny oraz relacji między obiektami.

Istotną zmianę w podejściu do przetwarzania obrazu przyniosła adaptacja architektury Transformer, pierwotnie dedykowanej przetwarzaniu języka naturalnego. Umożliwiło to rozwój modeli multimodalnych, które potrafią przetwarzać sygnał wizualny i tekstowy we wspólnej przestrzeni wektorowej. Rozwiązania te, reprezentowane przez architektury takie jak BLIP czy LLaVA, pozwalają na automatyzację zadań eksperckich, w tym generowanie opisów radiologicznych na podstawie zdjęć RTG czy ekstrakcję danych z dokumentacji technicznej.

Mimo potencjału dużych modeli fundamentowych, ich wdrożenie w specyficznych środowiskach produkcyjnych wiąże się z ograniczeniami technicznymi i prawnymi. Modele trenowane na ogólnych zbiorach danych często generują opisy nieprecyzyjne merytorycznie w dziedzinach wąskospecjalistycznych, takich jak diagnostyka obrazowa czy botanika. Dodatkowym wyzwaniem jest konieczność uzyskania danych wyjściowych w ustrukturyzowanych formatach, takich jak JSON lub CSV, co jest niezbędne do integracji systemów sztucznej inteligencji z bazami danych. Istotną barierą są również regulacje dotyczące prywatności, które często wymuszają przetwarzanie danych lokalnie, na sprzęcie o ograniczonych zasobach pamięciowych, co wyklucza użycie zewnętrznych interfejsów API.

W tym kontekście kluczowym zagadnieniem inżynierskim staje się opracowanie efektywnych metod adaptacji istniejących modeli otwartoźródłowych do specyficznych wymagań domenowych.

1.2. Cel pracy i pytania badawcze:

Głównym celem pracy jest zbadanie efektywności adaptacji modeli typu Vision Transformer do realizacji zaawansowanych zadań generowania opisów w domenach specjalistycznych. Praca ma charakter badawczo-wdrożeniowy i koncentruje się na empirycznej weryfikacji wpływu różnych strategii uczenia maszynowego na zdolność modelu do przyswajania wiedzy eksperckiej oraz formalnej.

Cel ten zostanie zrealizowany poprzez następujące działania szczegółowe:

– **Analiza porównawcza architektur:** Zbadanie wpływu rozmiaru oraz budowy modelu na jakość adaptacji do nowych zadań przy ograniczonym zbiorze treningowym.

– **Ewaluacja metod treningowych:** Porównanie klasycznego pełnego dostrajania, zwanego Full Fine-Tuning, z nowoczesnymi metodami efektywnymi parametrowo, takimi jak LoRA i QLoRA. Celem jest ustalenie, czy metody redukujące zapotrzebowanie na pamięć VRAM wiążą się z degradacją zdolności dyskryminacyjnych modelu w zadaniach wymagających wysokiej precyzji.

– **Weryfikacja zdolności strukturalnych:** Zbadanie możliwości nauczenia modelu wizyjnego roli parsera, czyli generowania poprawnych składniowo plików JSON bezpośrednio z obrazu, co stanowi alternatywę dla klasycznych potoków OCR.

– **Optymalizacja wydajności:** Określenie kompromisu pomiędzy czasem treningu i zużyciem zasobów sprzętowych a jakością końcową modelu.

1.3. Zakres pracy:

Praca obejmuje spektrum działań inżynierskich, począwszy od przygotowania danych, poprzez eksperymenty uczenia maszynowego, aż po wdrożenie rozwiązania w formie aplikacji.

Warstwa teoretyczna zawiera przegląd literatury dotyczącej ewolucji modeli językowo-wizyjnych, ze szczególnym uwzględnieniem zjawisk takich jak zapaść modalności wizyjnej, znana w literaturze jako Vision Token Collapse, oraz metod przeciwdziałania katastrofalnemu zapominaniu wiedzy.

W części badawczej wykorzystane zostaną wybrane modele dostępne na licencji Open Source. Eksperymenty zostaną przeprowadzone na autorskich lub specjalnie przygotowanych podzbiorach danych, reprezentujących dwa odmienne wyzwania: domenę medyczną lub przyrodniczą, gdzie kluczowa jest precyzja wizualna, oraz domenę inżynierską, wymagającą zachowania struktury logicznej danych wyjściowych. W ramach procedury badawczej zaimplementowane i przetestowane zostaną różne konfiguracje treningowe, uwzględniające techniki kwantyzacji oraz mieszanej precyzji obliczeń.

Zakres pracy w warstwie aplikacyjnej obejmuje zaprojektowanie i implementację prototypu systemu w architekturze klient-serwer. Część backendowa, oparta na języku Python i bibliotece PyTorch, będzie odpowiedzialna za inferencję modeli i obsługę żądań. Część frontendowa dostarczy interfejs graficzny umożliwiający użytkownikowi końcowemu interakcję z systemem, wybór modelu oraz wizualizację wyników w formie tekstowej lub ustrukturyzowanej.

2. Podstawy teoretyczne (Przegląd literatury)

2.1. Architektury multimodalne (Vision-Language Model)

Tradycyjne podejście do uczenia maszynowego traktowało analizę obrazu i przetwarzanie tekstu jako dwa odrębne problemy inżynieryjne, rozwiązywane za pomocą niekompatybilnych architektur. Systemy wizyjne, oparte głównie na splotowych sieciach neuronowych, specjalizowały się w ekstrahowaniu cech przestrzennych z macierzy pikseli w celu klasyfikacji obiektów (np. przypisania etykiety "samochód"). Z kolei systemy językowe skupiały się na analizie sekwencyjnej i modelowaniu gramatyki. Takie rozdzielenie uniemożliwiało tworzenie systemów zdolnych do pełnego zrozumienia sceny, w której treść wizualna jest nierozerwalnie związana z opisem semantycznym. Klasyfikacja obrazu daje jedynie zbiór etykiet, natomiast dopiero wygenerowanie opisu w języku naturalnym pozwala na uchwycenie relacji, akcji i atrybutów obiektów widocznych na zdjęciu.

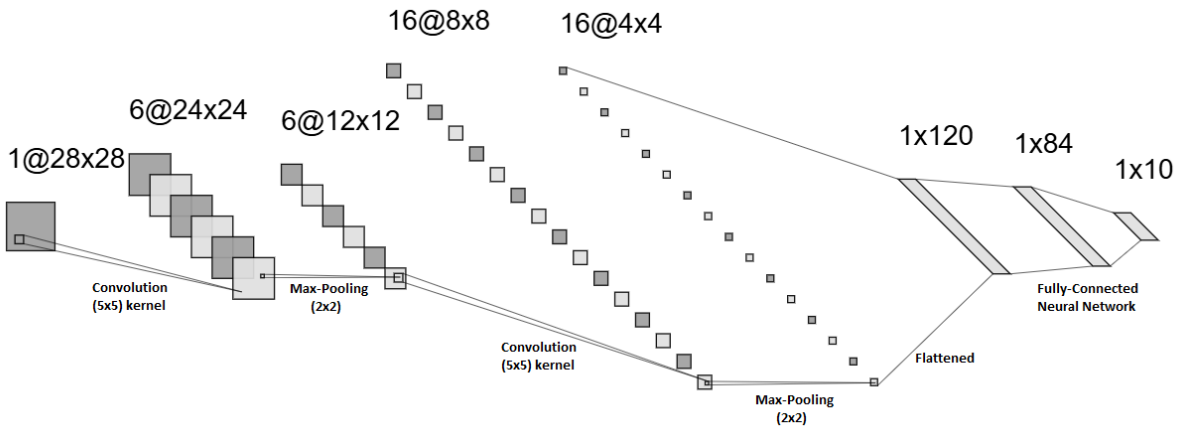
Rozwiązaniem tego problemu są modele multimodalne typu Vision-Language (VLM). Ich zadaniem jest integracja danych o skrajnie różnej strukturze: ciągłego sygnału wizualnego (wartości intensywności pikseli) oraz dyskretnego sygnału tekstowego (tokeny słownikowe). Wyzwanie inżynieryjne polega tutaj na stworzeniu mechanizmu, który pozwoli komputerowi "zrozumieć", że matematyczna reprezentacja obrazu psa jest tożsama z matematyczną reprezentacją słowa "pies". Modele VLM realizują to poprzez rzutowanie obu typów danych do wspólnej, wielowymiarowej przestrzeni wektorowej. W tej przestrzeni wektory reprezentujące obraz i odpowiadający mu tekst znajdują się blisko siebie, co pozwala na wykonywanie operacji logicznych łączących wzrok z językiem, takich jak generowanie podpisów czy wyszukiwanie obrazów za pomocą zapytań tekstowych.

2.1.1. Klasyczne podejście do generowania opisów: Architektury CNN-RNN

Do momentu spopularyzowania architektury Transformer (ok. 2020 roku), domyślnym standardem w zadaniach Image Captioning były architektury hybrydowe, łączące splotowe sieci neuronowe (CNN) z rekurencyjnymi sieciami neuronowymi (RNN). Podejście to opierało się na paradygmacie Encoder-Decoder, w którym dwie odrębne sieci współpracowały ze sobą w sekwencyjnym potoku przetwarzania.

CNN jako enkoder obrazu

Sieć konwolucyjna (CNN) pełniła rolę enkodera wizualnego, odpowiadając za ekstrakcję cech z obrazu wejściowego. Architektura CNN składa się z warstwy splotowej, która stosuje filtry do wykrywania lokalnych wzorców (np. krawędzi, tekstur), i warstwy poolingowej, która redukuje wymiarowość danych, zachowując najistotniejsze informacje. Warstwy te są wielokrotnie powtarzane, tworząc hierarchię cech o rosnącym poziomie abstrakcji. Na końcu sieci znajduje się warstwa w pełni połączona, której zadaniem jest przekształcenie wyekstrahowanych cech w wektor reprezentujący zawartość obrazu. Schemat przykładowej architektury CNN (LeNet-5) przedstawiono na rysunku 2.1.



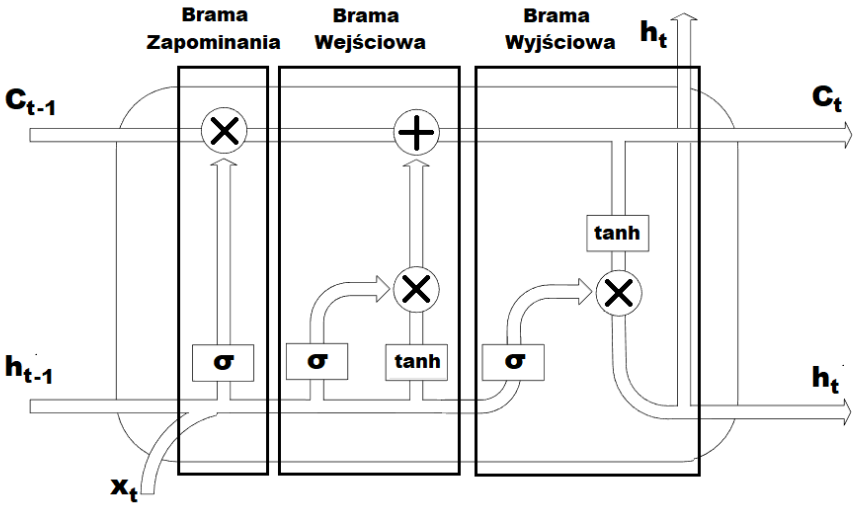
Rysunek 2.1: Schemat architektury sieci konwolucyjnej LeNet-5. Opracowanie własne.

Najważniejszym aspektem sieci CNN i równocześnie tym co odróżnia ją od klasycznych sieci w pełni połączonych, jest możliwość zachowania informacji przestrzennych obecnych w obrazie. W klasycznych sieciach wpełni połączonych, obraz jest przekształcany do jednowymiarowego wektora, co prowadzi do utraty informacji o lokalizacji cech. W przeciwieństwie do tego, CNN wykorzystuje operacje splotu i poolingu, które zachowują strukturę przestrzenną danych wejściowych. Dzięki temu sieci CNN są zdolne do lepszego rozpoznawania wzorców i obiektów w obrazach.

RNN jako dekoder tekstu

Role dekodera tekstu który generuje opis w języku naturalnym, pełniły rekurencyjne sieci neuronowe (RNN). Są to sieci w których wyjście z poprzedniego kroku czasowego jest wykorzystywane jako dodatkowe wejście do następnego kroku. Dzięki temu są w stanie zapamiętywać informacje z wcześniejszych stanów i używać ich do generowania kolejnych tokenów tekstowych. Zwykłae sieci RNN dobrze radzą sobie z krótkimi sekwencjami, jednak mają w przypadku dłuższych sekwencji pojawaiają się problemy z zanikiem gradientu, czyli trudnościami w uczeniu się długoterminowych zależności. Dlatego w praktyce dużo częściej stosowane były sieci LSTM (Long Short-Term Memory) lub GRU (Gated Recurrent Unit).

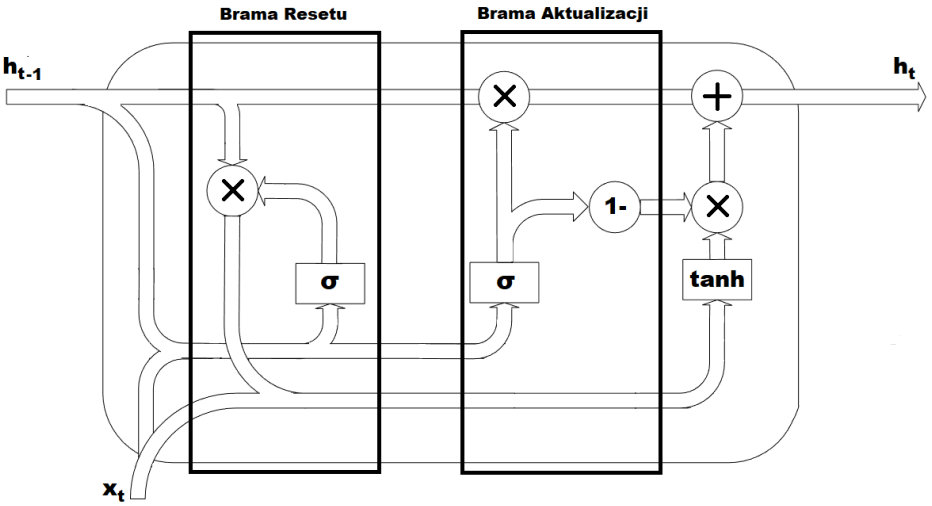
Sieć LSTM jest rozwinięciem architektury RNN, w przeciwieństwie do klasycznych sieci RNN, które w pojedynczym bloku posiadają jedną funkcję aktywacji, sieć LSTM zawiera trzy bramki: bramkę zapominania, bramkę wejścia i bramkę wyjścia. Struktura pojedynczej komórki sieci LSTM została przedstawiona na rysunku 2.2.



Rysunek 2.2: Schemat pojedynczej komórki sieci LSTM. Oznaczenia: σ – funkcja sigmoidalna, $\tanh$ – tangens hiperboliczny, $\times$ – mnożenie punktowe, $+$ – dodawanie punktowe. Opracowanie własne.

Brama zapominania decyduje które informacje z poprzedniego stanu komórki należy zachować, a które odrzucić. Zapominanie pozwala na usunięcie nieistotnych informacji, co pomaga w radzeniu sobie z długoterminowymi zależnościami. Brama wejścia określa które nowe informacje z bieżącego wejścia powinny zostać dodane do stanu komórki. Co pozwala na uczenie się nowych zależności. Brama wyjścia decyduje które informacje ze stanu komórki powinny zostać przekazane dalej. Dzięki tej strukturze sieci LSTM są w stanie efektywnie uczyć się długoterminowych zależności w sekwencjach danych, co czyni je idealnymi do zadań takich jak generowanie tekstu.

Alternatywą dla sieci LSTM, często stosowaną ze względu na mniejszą złożoność obliczeniową, jest sieć GRU (Gated Recurrent Unit). Wprowadzona jako uproszczenie mechanizmu LSTM, sieć GRU rezygnuje z oddzielnego stanu komórki, scalając go ze stanem ukrytym. W wyniku czego mamy tylko dwie bramki: bramkę resetującą i bramkę aktualizacji. Schemat pojedynczej jednostki GRU został przedstawiony na rysunku 2.3.



Rysunek 2.3: Schemat pojedynczej jednostki GRU. Oznaczenia: $1-$ – operacja dopełnienia do jedności, σ – funkcja sigmoidalna, $\tanh$ – tangens hiperboliczny. Opracowanie własne.

Bramka aktualizacji łączy w sobie funkcje bramki wejścia i zapominania znanych z LSTM. Decyduje ona, które informacje z poprzedniego stanu powinny zostać zachowane, a które zaktualizowane na podstawie bieżącego wejścia. Bramka resetu kontroluje, w jakim stopniu poprzedni stan powinien być brany pod uwagę przy generowaniu nowego kandydata na stan. Dzięki uproszczonej strukturze, sieci GRU są szybsze w treningu i zużywają mniej zasobów pamięciowych, często osiągając wyniki porównywalne do LSTM w zadaniach modelowania języka.

CNN-RNN

Integracja omówionych wcześniej komponentów – splotowego enkodera i rekurencyjnego dekodera – stanowiła fundament pierwszych skutecznych systemów typu Image Captioning, takich jak architektura "Show and Tell"[1]. W tym podejściu, obraz wejściowy jest najpierw przetwarzany przez sieć CNN pozbawioną ostatniej warstwy klasyfikacyjnej. W wyniku otrzymujemy wektor cech stanowiący numeryczną reprezentację obrazu. Wektor ten jest następnie przekazywany do sieci rekurencyjnej. Startując od specjalnego tokena startowego, sieć rekurencyjna generuje kolejne słowa opisu, bazując na wektorze cech obrazu oraz wcześniej wygenerowanych słowach. Proces ten kończy się po wygenerowaniu tokena końcowego. Architektura CNN-RNN umożliwiła znaczący postęp w zadaniach generowania opisów obrazów, ma ona jednak istotne ograniczenie znane jako "wąskie gardło informacyjne". Polega ono na tym, że cała informacja wizualna musi zostać skompresowana do pojedynczego wektora cech. Pomimo że jest to efektywniejsze podejście niż tradycyjne metody, wciąż prowadzi do utraty informacji o szczegółach i relacjach przestrzennych obecnych na obrazie. Dodatkowo fakt że dekoder widzi obraz jedynie na etapie inicjalizacji, prowadzi do tego żę generuje on kolejne słowa bazując jedynie na swojej zawodnej pamięci. Te ograniczenia doprowadziły do rozwoju mechanizmu uwagi (attention), a w rezultacie do powstania architektury Transformer, która zostanie szczegółowo omówiona w kolejnych sekcjach.

2.1.2. Mechanizm atencji i architektura Transformer

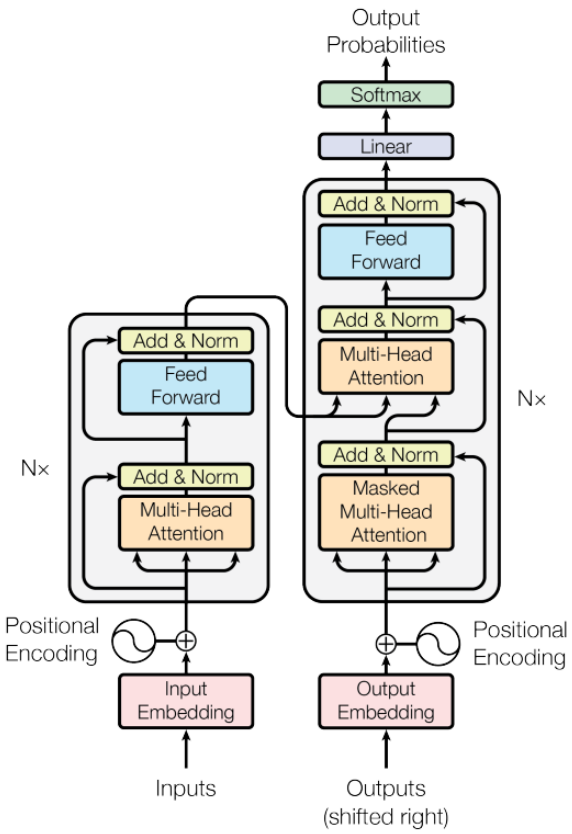
Kluczowym elementem nowoczesnych modeli językowych i multimodalnych jest mechanizm uwagi (attention). Pierwotnie wprowadzony w kontekście tłumaczenia maszynowego przez Bahdanau i in. [2], mechanizm ten rewolucjonizuje sposób, w jaki modele przetwarzają dane wejściowe, stając się odpowiedzią na problem "wąskiego gardła informacyjnego".

Mechanizm uwagi zamiast kompresowania całej informacji wejściowej do pojedynczego wektora, pozwala modelowi skupiać się na różnych częściach danych wejściowych w trakcie generowania każdego elementu wyjściowego. Oblicza on ważone sumy cech fragmentów wejściowych, dla każdej pozycji w sekwencji wyjściowej. Wagi te są dynamiczne i zależą od kontekstu na którym model aktualnie pracuje.

Architektura Transformer

Przełom w tym obszarze nastąpił w 2017 roku wraz z publikacją pracy "Attention is All You Need"[3], w której zaproponowano architekturę Transformer. Doprowadziło to do niemal całkowitego zastąpienia modeli rekurencyjnych na rzecz mechanizmu samoatencji (self-attention).

Architektura Transformer która została przedstawiona na rysunku 2.4, składa się z dwóch głównych komponentów: enkodera po lewej stronie i dekodera po prawej stronie. Enkoder odpowiada za przekształcenie danych wejściowych w ich reprezentację wektorową. Jego kluczowym zadaniem jest zrozumienie kontekstu i relacji pomiędzy wszystkimi elementami wejścia. Dekoder ma za zadanie generowanie sekwencji wyjściowej, korzystając z reprezentacji stworzonych przez enkoder oraz z tokenów wygenerowanych przez siebie w poprzednich krokach.



Rysunek 2.4: Architektura modelu Transformer. Źródło: Vaswani i in. [3].

Proces przetwarzania w architekturze Transformer zaczyna sie od tokenizacji danych wejściowych, które są następnie przekształcane w wektory osadzeń (embedding vectors). Wektory te są wzbogacane o informacje o pozycji (positional encoding), co pozwala modelowi uwzględniać kolejność elementów w sekwencji. Do enkodera przekazywane są kompletne dane wejściowe, natomiast dekoder otrzymuje sekwencyjnej tokenów wygenerowanych w poprzednich krokach. W enkoderze, dane przechodzą przez wielokrotne warstwy mechanizmu samoatencji oraz sieci feed-forward, co pozwala na uchwycenie złożonych zależności pomiędzy elementami. W dekoderze, dane przetwarzane są najpierw przez warstwy samoatencji z maskowaniem, a następnie wraz z danymi z enkodera przechodzą przez warstwy atencji krzyżowej (cross-attention) i sieci feed-forward. Na końcu dekodera znajduje się warstwa liniowa i funkcja softmax, które generują prawdopodobieństwa dla każdego tokena w słowniku. Dzięki temu że dane z enkodera są dostępne w każdym kroku dekodera, model może dynamicznie dostosowywać swoją uwagę do różnych części wejścia podczas generowania każdego tokena wyjściowego.

Samoatencja (Self-Attention)

Mechanizm samoatencji (self-attention) stanowi fundament architektury Transformera. Pozwala on modelowi na ocenę istotności różnych części sekwencji wejściowej względem siebie nawzajem. I w przeciwieństwie do tradycyjnych sieci rekurencyjnych, umożliwia równoległe przetwarzanie całej sekwencji. Pozwoliło to na znaczne przyspieszenie treningu i możliwość skalowania modeli do większych rozmiarów.

Z matematycznego punktu widzenia, mechanizm ten opiera się na trzech wektorach: wyznaczanych dla elementu wejściowego. Wejściem jest macierz osadzeń (embedding matrix), czyli macierz wektorów z których każdy wektor reprezentuje kolejne tokeny wejściowe. Dla każdego wektora w tej macierzy, obliczane są trzy wektory: zapytania (query), klucza (key) i wartości (value). Są one uzyskiwane poprzez mnożenie macierzy wejściowej przez trzy różne macierze wag, które są uczone podczas treningu. Obliczenie przykładowego wektora query dla

jednego z tokenów wejściowych wygląda tak:

$$q_i = x_i W^Q \tag{2.1}$$

Gdzie:

- q_i to wektor zapytania dla i-tego tokena wejściowego,
- x_i to wektor wejściowy dla i-tego tokena wejściowego,
- W^Q to macierz wag dla wektora zapytania.

Proces obliczania samoatencji, zwany również skalowaną atencją punktową (scaled dot-product attention), przebiega w kilku krokach. Najpierw obliczana jest macierz wyników dopasowania poprzez iloczyn skalarny wektorów zapytań i kluczy. Informuje to o tym, jak dobrze każdy token wejściowy pasuje do pozostałych tokenów. Następnie wyniki te są skalowane przez pierwiastek kwadratowy z wymiaru wektora klucza ($\sqrt{d_k}$), co pomaga w stabilizacji gradientów. Schemat dopasowania wraz z skalowaniem został przedstawiony na rysunku 2.5.

	$\mathbf{q}_1$	$\mathbf{q}_2$	$\mathbf{q}_3$	$\cdots$	$\mathbf{q}_n$
$\mathbf{k}_1$	$\frac{\mathbf{q}_1\mathbf{k}_1^T}{\sqrt{d_k}}$	$\frac{\mathbf{q}_2\mathbf{k}_1^T}{\sqrt{d_k}}$	$\frac{\mathbf{q}_3\mathbf{k}_1^T}{\sqrt{d_k}}$	$\cdots$	$\frac{\mathbf{q}_n\mathbf{k}_1^T}{\sqrt{d_k}}$
$\mathbf{k}_2$	$\frac{\mathbf{q}_1\mathbf{k}_2^T}{\sqrt{d_k}}$	$\frac{\mathbf{q}_2\mathbf{k}_2^T}{\sqrt{d_k}}$	$\frac{\mathbf{q}_3\mathbf{k}_2^T}{\sqrt{d_k}}$	$\cdots$	$\frac{\mathbf{q}_n\mathbf{k}_2^T}{\sqrt{d_k}}$
$\mathbf{k}_3$	$\frac{\mathbf{q}_1\mathbf{k}_3^T}{\sqrt{d_k}}$	$\frac{\mathbf{q}_2\mathbf{k}_3^T}{\sqrt{d_k}}$	$\frac{\mathbf{q}_3\mathbf{k}_3^T}{\sqrt{d_k}}$	$\cdots$	$\frac{\mathbf{q}_n\mathbf{k}_3^T}{\sqrt{d_k}}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\mathbf{k}_n$	$\frac{\mathbf{q}_1\mathbf{k}_n^T}{\sqrt{d_k}}$	$\frac{\mathbf{q}_2\mathbf{k}_n^T}{\sqrt{d_k}}$	$\frac{\mathbf{q}_3\mathbf{k}_n^T}{\sqrt{d_k}}$	$\cdots$	$\frac{\mathbf{q}_n\mathbf{k}_n^T}{\sqrt{d_k}}$

Rysunek 2.5: Rozszerzona macierz wyników dopasowania (Attention Scores).

Następnie na każdą kolumnę macierzy wyników dopasowania nakładana jest funkcja softmax, która przekształca wyniki w prawdopodobieństwa. Każdy element tego wektora reprezentuje wagę dopasowania danego tokena wejściowego względem tokena zapytania. Na koniec, obliczana jest ważona suma wektorów wartości, poprzez mnożenie tych wag z wektorami wartości v. Cały proces można zapisać równaniem:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.2}$$

Gdzie: Q, K i V to macierze zawierające odpowiednio zestawy wektorów zapytań, kluczy i wartości dla całej sekwencji. Otrzymany wynik to informacja jak należy zmienić początkowe reprezentacje wejściowe, aby uwzględnić kontekst całej sekwencji.

Wielogłowicowa atencja (Multi-Head Attention)

Pojedynczy mechanizm atencji może być ograniczony w swojej zdolności do uchwycenia różnych typów relacji jednocześnie. Dlatego stosujemy wiele "głów" atencji, każda z nich uczy się różnych aspektów zależności w danych wejściowych. Polega to na równoległym obliczaniu kilku mechanizmów atencji, przy czym każda głowa operuje na odrębnych podprzestrzeniach reprezentacji wektorowej. Każda głowa posiada własne macierze wag W_i^Q, W_i^K i W_i^V. Wyniki z poszczególnych głów są następnie łączone i przekształcane za pomocą dodatkowej macierzy wag W^O. Dzięki temu model może jednocześnie skupiać się na różnych aspektach danych wejściowych.

Atencja z maskowaniem (Masked Attention) i Atencja krzyżowa (Cross-Attention)

W architekturze Transformera, pojawia sie jeszcze atencja z maskowaniem i atencja krzyżowa. Atencja z maskowaniem jest stosowana w dekoderze, aby zapobiec "podglądaniu"przyszłych tokenów podczas generowania sekwencji wyjściowej. Oznacza to że na dany token wpływ mogą mieć tylko tokeny wcześniejsze lub bieżące. W praktyce oznacza to, że podczas obliczania wag atencji, elementy odpowiadające przyszłym tokenom są maskowane (ustawiane na bardzoniską wartość, np. $-\infty$), dzięki temu po zastosowaniu funkcji softmax, otrzymują one wagę bliską zeru co zostało przedstawione na rysunku 2.6.

	$\mathbf{q}_1$	$\mathbf{q}_2$	$\mathbf{q}_3$	$\dots$	$\mathbf{q}_n$
$\mathbf{k}_1$	$\frac{\mathbf{q}_1\mathbf{k}_1^T}{\sqrt{d_k}}$	$\frac{\mathbf{q}_2\mathbf{k}_1^T}{\sqrt{d_k}}$	$\frac{\mathbf{q}_3\mathbf{k}_1^T}{\sqrt{d_k}}$	$\dots$	$\frac{\mathbf{q}_n\mathbf{k}_1^T}{\sqrt{d_k}}$
$\mathbf{k}_2$	$-\infty$	$\frac{\mathbf{q}_2\mathbf{k}_2^T}{\sqrt{d_k}}$	$\frac{\mathbf{q}_3\mathbf{k}_2^T}{\sqrt{d_k}}$	$\dots$	$\frac{\mathbf{q}_n\mathbf{k}_2^T}{\sqrt{d_k}}$
$\mathbf{k}_3$	$-\infty$	$-\infty$	$\frac{\mathbf{q}_3\mathbf{k}_3^T}{\sqrt{d_k}}$	$\dots$	$\frac{\mathbf{q}_n\mathbf{k}_3^T}{\sqrt{d_k}}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\mathbf{k}_n$	$-\infty$	$-\infty$	$-\infty$	$\dots$	$\frac{\mathbf{q}_n\mathbf{k}_n^T}{\sqrt{d_k}}$

Rysunek 2.6: Macierz atencji z maskowaniem (Masked Attention), uniemożliwiająca dostęp do przyszłych tokenów.

Z kolei atencja krzyżowa jest stosowana w dekoderze do integracji informacji z enkodera. Różnica polega na tym, że wektory zapytań (Q) pochodzą z dekodera, natomiast wektory kluczy (K) i wartości (V) pochodzą z enkodera. Pozwala to dekoderowi na dynamiczne dostosowywanie swojej uwagi do różnych części wejścia podczas generowania następnych tokenów wyjściowych.

2.1.3. Vision Transformer (ViT)

Początkowo architektura transformera wykorzystywana była głównie w zadaniach przetwarzania języka naturalnego (NLP). Uważano, że do analizy danych wizualnych niezbędne są operacje splotu. Paradygmat ten uległ zmianie w 2020 roku wraz z publikacją pracy “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale” [4]. W pracy tej zaproponowano model Vision Transformer (ViT), który zrywa z koncepcją przetwarzania obrazu jako dwuwymiarowej siatki pikseli na rzecz traktowania go jako sekwencji. Podejście to polega na podzieleniu obrazu na siatkę mniejszych, fragmentów (ang. *patches*), a następnie przekształcenie ich do postaci wektorów. Z perspektywy modelu obraz staje się strukturą analogiczną do zdania podzielonego na tokeny.

Architektura Vision Transformer posiada znacznie mniejsze wbudowane obciążenie indukcyjne (ang. *inductive bias*) niż sieci CNN. Model nie posiada wiedzy o dwuwymiarowej strukturze obrazu ani o relacjach sąsiedztwa pikseli. Te relacje muszą zostać wyuczone od podstaw w procesie treningu, oznacza to że aby osiągnąć efekty porównywalne lub lepsze od sieci splotowych, modele ViT wymagają zazwyczaj znacznie większych zbiorów danych treningowych.

Przetwarzanie obrazu w Vision Transformer

Standardowo modele transformer przyjmują na wejściu sekwencję jednowymiarowych wektorów. Aby dostosowac obraz o wymiarach $H \times W \times C$ (wysokość, szerokość, liczba kanałów) do tego formatu, obraz jest dzielony na siatkę N fragmentów obliczonych jako:

$$N = \frac{H \times W}{P^2} \tag{2.3}$$

Gdzie P to rozmiar boku kwadratowego fragmentu (patch). Te fragmenty są następnie przekształcane do jednowymiarowych wektorów poprzez spłaszczenie i rzutowane liniowo.

Do każdego wektora dodawany jest wektor pozycyjny (positional embedding), który koduje informacje o oryginalnej lokalizacji fragmentu w obrazie. Dodatkowo do całej sekwencji dodawany jest specjalny token klasyfikacyjny (CLS token), którego stan końcowy jest reprezentacją całego obrazu. Proces przygotowania wektora wejściowego z_0 można opisać równaniem:

$$z_0 = [x_{class}; x_p^1 E; x_p^2 E; \ldots; x_p^N E] + E_{pos} \tag{2.4}$$

Gdzie:

- x_{class} to wektor specjalnego tokena klasyfikacyjnego,
- x_p^i to i-ty fragment obrazu,
- E to macierz wag do rzutowania fragmentów na przestrzeń wektorową,
- E_{pos} to macierz osadzeń pozycyjnych.

Architektura Enkodera

Pomimo że architektura Vision Transformer jest bardzo podobna do standardowego transformera, istnieje modyfikacja znana jako architektura Pre-Norm. W orginalnym transformerze, normalizacja warstw następuje po mechanizmie atencji i sieci MLP. W modelu ViT normalizacja jest stosowana prze każdym z tych komponentów. Ta zmiana poprawia stabilność treningu i pozwala na trenowanie głębszych modeli. Dodatkową różnicą jest stosowanie GELU (Gaussian Error Linear Unit) zamiast ReLU jako funkcji aktywacji w sieci MLP.Poza tymi modyfikacjami, mechanizm przetwarzania pozostaje niezmieniony.

2.1.4. Architektua VLM

Współczesne modele generowania opisów obrazów odchodzą od dedykowanych dekoderów rekurencyjnych na rzecz potężnych pretrenowanych Dużych Modeli Językowych (LLM). Pełny model do opisywania obrazów składa się z wizyjnego enkodera (np. Vision Transformer), dużego modelu językowego oraz modułu dopasowania który integruje oba komponenty. Bezpośrednie połączenie Visual Transformer z LLM jest często niemożliwe ze względu na różnice w wymiarach przestrzeni wektorowych obu modeli. Dlatego stosuje się dodatkowy moduł dopasowania (adapter), jego zadaniem jest rzutowanie wyjścia enkodera wizualnego do przestrzeni wejściowej LLM. W najprostszej postaci jest to prosta sieć MLP lub warstwa liniowa. Proces ten można opisać wzorem:

$$H_v = W \times Z_{img} + b \tag{2.5}$$

Gdzie Z_{img} to wyjście enkodera wizualnego, W i b to wagi i bias modułu dopasowania, a H_v to wynikowa sekwencja tokenów wizualnych. Z perspektywy LLM, tokeny te są traktowane tak samo jak wektory słów.

Pełny przepływ operacji w modelu VLM do generowania opisów obrazów zaczyna się od przetworzenia obrazu przez enkoder ViT, otrzymana reprezentacja jest transformowana przez adapter do formatu zgodnego z LLM. Następnie, LLM generuje opis w języku naturalnym, korzystając z tokenów wizualnych jako kontekstu.

Zaletami modeli VLM są:

– Zdolność wnioskowania i wiedza ogólna: Dzięki wykorzystaniu LLM, modele VLM nie są ograniczone do słownictwa występującego w zbiorze treningowym, lecz mogą dostarczać kontekst którego nie ma na obrazie.

– Generalizacja Zero-Shot: Modele VLM potrafią generować opisy dla obrazów spoza zbioru treningowego, o ile obiekty te są znane LLM.

– Sterowalność: W przeciwieństwie do modeli RNN, modele VLM pozwalają na sterowanie formatowaniem i treścią wyjścia za pomocą promptów tekstowych.

2.2. Wyzwania w uczeniu modeli multimodalnych

Proces uczenia modeli multimodalnych typu Vision-Language napotyka na szereg unikalnych wyzwań, które wynikają z konieczności integracji i przetwarzania danych o różnej naturze. W niniejszym podrozdziale omówiono kluczowe wyzwania na które można natknąć się podczas trenowania takich modeli.

2.2.1. Zapaść modalności (Vision Token Collapse)

Zapaść modalności (ang. *Vision Token Collapse*) definiowana jest jako zjawisko, w którym komponent językowy (LLM) zaczyna ignorować informacje pochodzące z komponentu wizualnego (enkodera obrazu). W rezultacie model generuje opisy opierając się wyłącznie na wiedzy zawartej w wagach LLM, pomijając fakty wizualne obecne na obrazie.

Zjawisko to występuje najczęściej w sytuacji, gdy silny dekoder tekstowy zostaje zparowany z relatywnie słabym enkoderem wizualnym. Jedną z głównych przyczyn tego zjawiska jest dominacja modalności tekstowej. Wynika to z faktu, żę w trakcie treningu modele językowe są trenowane na danych o rzędy wielkości większych niż dane obraz-tekst. Podczas treningu gradientu powoduje to silniejszą optymalizację pod kontem poprawności gramatycznej i semantycznej niż pod kątem interpretacji wizualnej. Innym powodem może być luka modalności (ang. *modality gap*), która jest powodowana tym, że nawet po rzutowaniu do wspólnej przestrzeni wektorowej, reprezentacji wizualnych i tekstowych pozostają one odseparowane geometrycznie [5]. Jeśli ta luka nie zostanie zredukowana przez adapter, model nauczy się ignorować tokeny wizualne aby minimalizować błąd predykcji.

Skutkiem zapaści modalności są tzw. halucynacje wizualne (ang. *visual hallucinations*), model generuje opisy które są poprawne językowo, ale niezgodne z faktyczną zawartością obrazu. W zastosowaniach specjalistycznych, takich jak medycyna czy inżynieria, jest to błąd krytyczny, dyskwalifikujący model z użycia produkcyjnego. Przeciwdziałanie zjawisku zapaści modalności często wymusza stosowanie zaawansowanych technik treningowych, takich jak zamrażanie wag LLM w początkowych etapach treningu.

2.2.2. Katastrofalne zapominanie (Catastrophic Forgetting)

Kolejnym wyzwaniem w trenowaniu modeli multimodalnych jest zjawisko katastrofalnego zapominania (ang. *catastrophic forgetting*). Polega ono na tym, że podczas aktualizacji parametrów modelu pod kątem nowych danych, model traci zdolność do wykonywania wcześniej nauczonych zadań. Problem ten wynika z tzw. dylematu stabilności-plastyczności (ang. *stability-plasticity dilemma*), system musi być wystarzająco plastyczny, aby móc uczyć się nowych informacji, ale równocześnie wystarczająco stabilny, aby nie zapominać wcześniej nabytych umiejętności.

W przypadku klasycznego pełnego dostrajania (Full Fine-Tuning), aktualizowane są wszystkie wagi sieci dążąc do zminimalizowania błędu na nowych danych. Prowadzi to do nadpisania wag, które w procesie pre-treningu kodowały ogólną wiedzę o świecie. Kirkpatrick i in. [6] wykazali, że wagi kluczowe dla poprzednich zadań są często drastycznie zmieniane, powodując degradację wydajności modelu.

W modelach VLM katastrofalne zapominanie objawia się na dwa sposoby:

– Utrata wiedzy językowej: Model traci płynność i poprawność językową.

– Utrata zdolności wizualnej: Model w wyniku dostrajania dla danych z wąskiej dziedziny traci zdolność do rozpoznawania obiektów powszechnych.

Aby temu zapobiegać stosuje się metodę uczenia przyrostowego (ang. Continual Learning), lub techniki efektywne parametrowo (PEFT), takie jak LoRA. Jak wskazują Hu i in. [7], metody te pozwalają na dostosowanie modelu do nowych zadań bez konieczności modyfikowania wszystkich wag sieci, co pomaga w zachowaniu wcześniej nabytej wiedzy. Ze względu na kluczową rolę technik PEFT w niniejszej pracy, ich szczegółowa charakterystyka oraz architektura zostały omówione w kolejnym podrozdziale.

2.3. Metody adaptacji (Fine-tuning)

Adaptacja modeli w paradygmacie transferu uczenia (ang. *transfer learning*), polega na douczaniu pretrenowanego modelu na nowych, często wyspecjalizowanych zbiorach danych. Celem jest dostosowanie modelu do konkretnego zadania wykorzystując jego wcześniej nabyte umiejętności. Wybór strategi adaptacji sprowadza się do kompromisu pomiędzy dostępnymi zasobami obliczeniowymi, a jakością adaptacji i ryzykiem wystąpienia zjawiska katastrofalnego zapominania. W tym podrozdziale omówiono dwie główne metody adaptacji: pełne dostrajanie (Full Fine-Tuning) oraz techniki efektywne parametrowo (Parameter-Efficient Fine-Tuning, PEFT).

2.3.1. Pełne dostrajanie (Full Fine-Tuning, FFT)

Klasycznym podejściem do adaptacji pretrenowanych modeli jest pełne dostrajanie, w którym procesowi optymalizacji poddawane są wszystkie trenowalne parametry modelu. W tym scenariuszu, model zainicjalizowany pretrenowanymi wagami jest trenowany przy użyciu standardowego algorytmu propagacji wstecznej (backpropagation) [8].

Główną zaletą tego podejścia jest wysoka elastyczność. Dzięki możliwości aktualizacji wszystkich wag, model posiada największy potencjał do uczenia skomplikowanych wzorców i adaptacji do specjallistycznego słownictwa. Zakładając, że dysponujemy odpowiednio dużym zbiorem danych, oraz zasobami obliczeniowymi, metoda ta zazwyczaj pozwala osiągnąć najlepszą jakość predykcji.

Niemniej jednak, pełne dostrajanie wiąże się z istotnymi wyzwaniami. Zapotrzebowanie na pamięć podczas treningu rośnie nie tylko ze względu na konieczność przechowywania wszystkich wag, ale również ich gradientów i stanów optymalizatora. Szacuje się, że dla modelu o rozmiarze 7 miliardów parametrów, pełne dostrajanie w precyzji 16-bitowej wymaga ponad 100 GB pamięci VRAM, co wyklucza użycie konsumenckiego sprzętu GPU. Dodatkowo występuje ryzyko katastrofalnego zapominania, gdzie model traci wcześniej nabyte umiejętności. Występuje także ryzyko nadmiernego dopasowania (overfittingu). Gdzie przy relatywnie małych zbiorach danych, modyfikacja wszystkich parametrów prowadzi do zapamiętania danych treningowych tracąc zdolność do generalizacji dla nowych przykładów.

2.3.2. Techniki efektywne parametrowo (Parameter-Efficient Fine-Tuning, PEFT)

W odpowiedzi na wyzwania związane z pełnym dostrajaniem, nastąpił rozwój technik efektywnych parametrowo (PEFT). Polega to na zamrożeniu większości parametrów pretrenowanego modelu i ograniczeniu procesu uczenia do niewielkiego podzbioru wag dodatkowych [9]. W porównaniu do FFT, liczba parametrów poddawanych aktualizacji jest rzędy wielkości mniejsza, często stanowiąc zaledwie 0.1% do 1% całkowitej liczby parametrów modelu.

Zastosowanie technik PEFT przynosi korzyści związane z redukcją zapotrzebowania na pamięć. Ponieważ wynikiem treningu jest lekki plik z wagami adaptacyjnymi, pozwala to na utrzymanie jednej instancji modelu bazowego i dynamiczne ładowanie różnych zestawów wag adaptacyjnych dla różnych zadań. Dodatkowo dzięki zamrożeniu oryginalnych wag, model zachowuje swoją wiedzę ogólną, co redukuje ryzyko katastrofalnego zapominania.

Low-Rank Adaptation (LoRA)

Metoda Low-Rank Adaptation (LoRA), zaproponowana przez Hu i in. [7], opiera się na hipotezie o "niskim wymiarze wewnętrznym" (ang. *intrinsic low dimension*) zmian wag sieci podczas dostrajania. Autorzy zauważyli, że aktualizacje niezbędne do przystosowania modelu do nowego zadania mażna reprezentować w przestrzeni o znacznie niższej rzędowości niż oryginalne wagi.

Dla danej macierzy wag $W_0 \in \mathbb{R}^{d\times k}$, zamiast uczyć całą tą macierz aktualizacji ΔW, LoRA wprowadza dekompozycję na iloczyn dwóch mniejszych macierzy B i A:

$$W_0 + \Delta W = W_0 + BA \tag{2.6}$$

Gdzie: $A \in \mathbb{R}^{r\times k}$ oraz $B \in \mathbb{R}^{d\times r}$ natomiast r to rząd niskiego wymiaru, gdzie $r \ll \min(d, k)$.

Podczas treningu, oryginalna macierz wag W_0 pozostaje zamrożona, aktualizowane są jedynie macierze A i B. Ponieważ r przyjmuje bardzo małe wartości (np. 8 lub 16), liczba trenowanych parametrów drastycznie spada.

Ważnym aspektem jest inicjalizacja macierzy A i B: macierz A inicjalizowana jest losowym rozkładem gaussowskim, natomiast macierz B jest inicjalizowana zerami. Gwarantuje to, że na początku treningu $\Delta W = 0$ a model zachowuje się jak oryginalny pretrenowany model.

Istotną zaletą LoRA jest brak opóźnień podczas inferencji. Po treningu macierze A i B mogą być ze sobą pomnożone i dodane do oryginalnych wag W_0. Dzięki temu otrzymujemy standardowy model który nie wymaga dodatkowych obliczeń w czasie działania.

Quantized Low-Rank Adaptation (QLoRA)

Quantized Low-Rank Adaptation (QLoRA) to rozszerzenie metody LoRA opracowane przez Dettmersa i in. [10]. Ta technika ukierunkowana jest na maksymalną redukcję zapotrzebowania pamięci.

Kluczowym aspektem do zrozumienia QLoRA jest pojęcie kwantyzacji (ang. *quantization*). Jest to proces dyskretyzacji parametrów modelu, polegający na mapowaniu ciągłych wartości o wysokiej precyzji (np. 16-bitowych liczb zmiennoprzecinkowych) do wartości dyskretnych niższej precyzji (np. 4-bitowych liczb całkowitych).

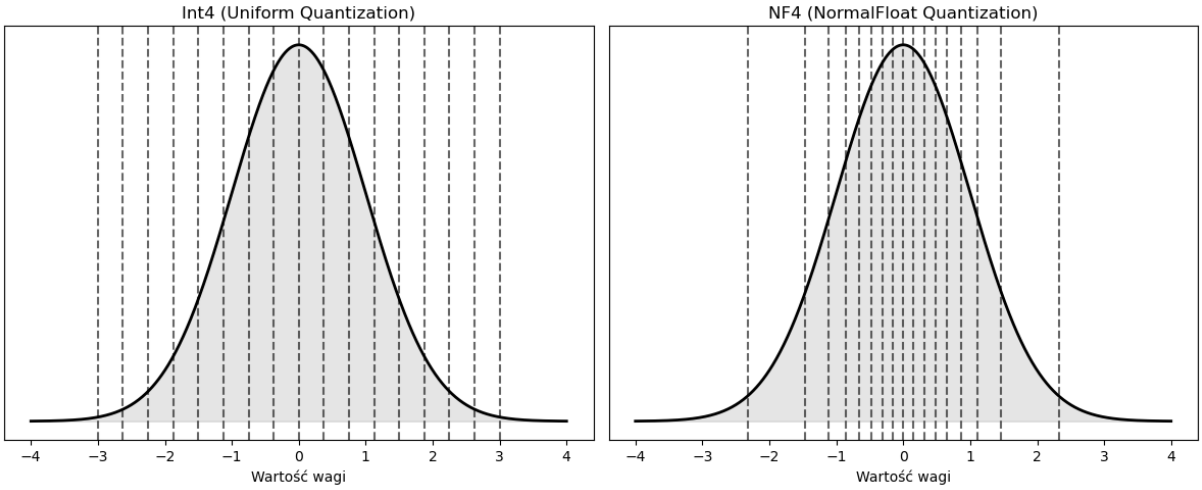
Proces ten to rzytowanie wagi w do najbliższej wartości reprezentowalnej w niższej precyzji:

$$w_q = \text{round}\left(\frac{w}{S}\right) + \text{Z} \tag{2.7}$$

Gdzie: S to skala kwantyzacji, a Z to przesunięcie (zero-point).

Metoda QLoRA pozwala na efektywne trenowanie modeli skwantyzowanych do niskiej precyzji (np. 4-bitowej) poprzez zastosowanie LoRA na skwantyzowanych wagach. Choć kwantyzacja drastycznie zmniejsza rozmiar modelu, wiąże się z wprowadzeniem błędu kwantyzacji który może negatywnie wpływać na jakość predykcji. Głównym wyzwaniem jest fakt, że standardowe typy danych (np. int4) nie radzą sobie z reprezentacją wag sieci.

Aby rozwiązać ten problem stosuje się NF4 (normal float 4-bit), czyli specjalny typ danych. Autorzy metody zauważyli, że rozkład wag w modelach językowych jest zbliżony do rozkładu normalnego. Typ NF4 przydziela więcej przedziałów wartościom bliskim zera gdzie znajduje się większość wag. Podział ten przedstawia rysunek 2.7. Minimalizuje to błąd kwantyzacji.



Rysunek 2.7: Dopasowanie przedziałów kwantyzacji do rozkładu normalnego wag w metodach Int4 (lewa strona) oraz NF4 (prawa strona). Źródło: Opracowanie własne.

Dodatkowo, QLoRA wykorzystuje podwójne kwantyzowanie (double quantization). Aby odzyskać precyzję wagi posiadają stałą skalującą S. Jednak przechowywanie tej skali dla każdej wagi jest kosztowne pamięciowo. Dlatego QLoRA poddaje kwantyzacji również stałe skalujące.

W architekturze QLoRA, model bazowy jest przechowywany w formacie NF4, podczas gdy warstwy LoRA są przechowywane w standardowej precyzji. Podczas obliczeń wagi są dekwantyzowane, a następnie stosowane są aktualizacje LoRA.

2.4. Metody beztreningowe i kontekstowe

Alternatywą dla adaptacji modeli poprzez trening są metody beztreningowe, które wykorzystują zdolności dużych modeli językowych do uczenia się na podstawie kontekstu. Podejście to zakłada, że model posiada wystarczającą wiedzę ogólną, a jego adaptacja do nowego zadania odbywa sie wyłącznie poprzez manipulację wejściem (prompting) lub dostarczenie zewnętrznego kontekstu.

2.4.1. Uczenie w kontekście (In-Context Learning, ICL)

Metoda uczenia w kontekście, szczegółowo przeanalizowana w pracy Browna i in. [11], polega na dostarczeniu modelowi instrukcji oraz przykładów w ramach promptu wejściowego. W zadaniach generowania opisów obrazów, model otrzymuje kilka par obraz-opis, a na końcu obraz docelowy z prośbą o wygenerowanie opisu.

Główną zaletą ICL jest brak kosztów treningowych i natychmiastowa adaptacja. Jednakże w zadaniach specjalistycznego opisywania obrazów, metoda ta napotyka na istotne bariery:

– Ograniczone okno kontekstowe: Modele LLM mają ograniczoną długość wejścia, co ogranicza liczbę przykładów które można dostarczyć.

– Brak głębokiej adaptacji: ICL pozwala modelowi naśladować styl lub format odpowiedzi, ale nie jest wystarczający do nauczenia modelu rozpoznawania nowych cech wizualnych, których model nie widział w czasie pre-treningu.

2.4.2. Generowanie wspomagane wyszukiwaniem (Visual RAG)

Poejście RAG (Retrieval-Augmented Generation) [12]. Metoda ta polega na wyposażeniu modelu w dostęp do zewnętrznej bazy wiedzy, z której może czerpać informacje podczas generowania odpowiedzi.

Model najpierw wyszukuje obrazy podobne do obrazu wejściowego w załączonej bazie danych, a następnie dołącza ich opisy do promptu wejściowego jako kontekst pomocniczy.

Pomimo potencjału tej metody, wprowadza ona znaczną złożoność architektonizną systemu, wymagając implementacji efektywnego mechanizmu wyszukiwania oraz utrzymania bazy danych. Dodatkowo przez konieczność szukania i dołączania dodatkowych informacji, czas odpowiedzi się zwiększa.

3. Metodyka badań i narzędzia

4. Eksperymenty i analiza wyników

5. Projekt i implementacja aplikacji

6. Dyskusja i Podsumowanie

Bibliografia

[1] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. Prezentowane na ICLR 2015.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, volume 30, 2017.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[5] Weixin Liang, Yuhui Zhang, Yongchan Kwon, Serena Yeung, and James Zou. Mind the gap: Understanding the modality gap in multi-modal contrastive representation learning. In *Advances in Neural Information Processing Systems*, volume 35, pages 17612–17625, 2022.

[6] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[7] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[8] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[9] Vladislav Lialin, Vijeta Deshpande, and Anna Rumshisky. Scaling down to scale up: A guide to parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.15647*, 2023.

[10] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems*, volume 36, 2024.

[11] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in neural information processing systems*, volume 33, pages 1877–1901, 2020.

[12] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474, 2020.