Kompresja sygnałów EKG z wykorzystaniem autoenkodera

Bartłomiej Rydzak, Mateusz Adamczyk, Michał Saturczak

17 czerwca 2025

Spis treści

1 Wprowadzenie

Niniejszy projekt przedstawia implementację systemu kompresji sygnałów elektrokardiograficznych (EKG) z wykorzystaniem głębokich sieci neuronowych. W przeciwieństwie do tradycyjnych metod kompresji, zastosowane rozwiązanie wykorzystuje nowoczesne narzędzia uczenia maszynowego - TensorFlow 2.x oraz Keras API - do stworzenia autoenkodera zdolnego do stratnej kompresji sygnałów biologicznych.

Głównym celem pracy było opracowanie efektywnego mechanizmu redukcji wymiarowości sygnałów EKG przy zachowaniu kluczowych charakterystyk diagnostycznych. Projekt stanowi praktyczne zastosowanie teorii sieci neuronowych w dziedzinie przetwarzania sygnałów biomedycznych.

2 Fundamenty teoretyczne

2.1 Autoenkodera w kontekście kompresji

Autoenkoder stanowi szczególny typ architektury sieci neuronowej, której zadaniem jest nauczenie się efektywnej reprezentacji danych wejściowych w przestrzeni o zmniejszonej wymiarowości. Struktura składa się z dwóch głównych komponentów:

- **Enkoder** $f_\theta : \mathbb{R}^n \to \mathbb{R}^k$ - funkcja mapująca dane wejściowe do reprezentacji ukrytej
- **Dekoder** $g_\phi : \mathbb{R}^k \to \mathbb{R}^n$ - funkcja rekonstruująca dane z reprezentacji ukrytej

gdzie $k < n$ określa stopień kompresji, a θ, ϕ reprezentują parametry uczenia odpowiednich części sieci.

2.2 Funkcja straty i optymalizacja

Proces uczenia autoenkodera opiera się na minimalizacji funkcji straty rekonstrukcji:

$$\mathcal{L}(\theta, \phi) = \frac{1}{m} \sum_{i=1}^{m} ||x^{(i)} - g_\phi(f_\theta(x^{(i)}))||^2 \tag{1}$$

gdzie m oznacza liczbę próbek treningowych, a $x^{(i)}$ reprezentuje i-tą próbkę sygnału EKG.

2.3 Nieliniowe transformacje sygnału

W celu lepszego wykorzystania zakresu dynamicznego sieci, zastosowano nieliniowe transformacje:

- **Przed enkodowaniem**: $x_{transformed} = \sqrt{x_{normalized}}$
- **Po dekodowaniu**: $x_{reconstructed} = (decoder_{output})^2$

Transformacje te pozwalają na lepsze mapowanie charakterystyk sygnałów EKG, które często zawierają wartości w wąskim zakresie z pojedynczymi skokami amplitudy.

3 Zbiór danych i preprocessing

3.1 Charakterystyka datasetu

Projekt wykorzystuje zbiór danych "ECG Heartbeat Categorization"dostępny na platformie Kaggle, składający się z dwóch głównych baz:

Zbiór danych	Liczba próbek	Zastosowanie
MIT-BIH Arrhythmia	∼87,000	Trening i walidacja
PTB Diagnostic ECG	∼14,500	Testy dodatkowe

Tabela 1: Charakterystyka wykorzystanych zbiorów danych EKG

3.2 Pipeline preprocessing

Proces przygotowania danych obejmuje następujące etapy:

1. **Filtracja kategorii**: Ekstrakacja sygnałów normalnych (klasa 0) dla uczenia nienadzorowanego
2. **Normalizacja min-max**: Skalowanie wartości do zakresu $[0, 1]$
3. **Konwersja typów**: Rzutowanie na `float32` dla optymalizacji obliczeń
4. **Transformacja nieliniowa**: Stosowanie pierwiastkowania przed enkodowaniem

```
def normalize(train_arr, test_arr):
    min_val = np.min(train_arr, axis=(0,1))
    max_val = np.max(train_arr, axis=(0,1))
    train_norm = (train_arr - min_val) / (max_val - min_val)
    test_norm = (test_arr - min_val) / (max_val - min_val)
    return train_norm, test_norm
```

Listing 1: Funkcja normalizacji danych

4 Architektura systemu

4.1 Specyfikacja modelu

Zaprojektowany autoenkoder charakteryzuje się symetryczną architekturą z wąskim gardłem (bottleneck) w warstwie środkowej:

Warstwa	Wymiar wej.	Wymiar wyj.	Aktywacja
	Enkoder		
Dense 1	187	100	ReLU
Dense 2	100	40	ReLU
Dense 3 (Bottleneck)	40	20	Linear
	Dekoder		
Dense 4	20	40	ReLU
Dense 5	40	100	ReLU
Dense 6 (Output)	100	187	Sigmoid

Tabela 2: Architektura sieci neuronowej autoenkodera

4.2 Implementacja w TensorFlow/Keras

Model zaimplementowano wykorzystując nowoczesne API Keras z podklasowaniem `tf.keras.Model`:

```
class ECGAutoEncoder(tf.keras.Model):
    def __init__(self):
        super(ECGAutoEncoder, self).__init__()

        self.encoder = tf.keras.Sequential([
            tf.keras.layers.Dense(100, activation='relu',
               input_shape=(188,)),
            tf.keras.layers.Dense(40, activation='relu'),
            tf.keras.layers.Dense(20, activation='linear')
        ])

        self.decoder = tf.keras.Sequential([
            tf.keras.layers.Dense(40, activation='relu',
               input_shape=(20,)),
            tf.keras.layers.Dense(100, activation='relu'),
            tf.keras.layers.Dense(188, activation='sigmoid')
        ])

    def call(self, x):
        encoded = self.encode(x)
        decoded = self.decode(encoded)
        return decoded

    def encode(self, x):
        return self.encoder(tf.sqrt(x))

    def decode(self, encoded):
        return tf.square(self.decoder(encoded))
```

Listing 2: Implementacja klasy ECGAutoEncoder

4.3 Parametry kompresji

System osiąga **współczynnik kompresji 9.35:1**, redukując wymiarowość sygnału z 187 do 20 składowych. Ta znacząca redukcja wymiarów pozwala na efektywne przechowywanie i transmisję sygnałów EKG przy zachowaniu kluczowych informacji diagnostycznych.

5 Proces uczenia

5.1 Konfiguracja treningu

Proces uczenia został skonfigurowany z następującymi parametrami:

- **Optymalizator**: Adam z learning rate 1×10^{-3}
- **Funkcja straty**: Mean Squared Error (MSE)
- **Rozmiar batcha**: 250 próbek na epokę
- **Maksymalna liczba epok**: 200
- **Early stopping**: 15 epok bez poprawy

5.2 Strategia early stopping

Zaimplementowano zaawansowany mechanizm wczesnego zatrzymania oparty na monitorowaniu:

- Docelowego RMSE poniżej 0.0086
- Stagnacji poprawy poniżej 1×10^{-6} przez 15 epok
- Zachowania najlepszych wag podczas procesu uczenia

```
def train_epoch(model, optimizer, dataset, val_data, loss_fn):
    train_ds = tf.data.Dataset.from_tensor_slices(dataset).shuffle
       (1000).batch(250)

    for batch in train_ds:
        with tf.GradientTape() as tape:
            recon = model(batch)
            loss = loss_fn(batch, recon)
        grads = tape.gradient(loss, model.trainable_variables)
        optimizer.apply_gradients(zip(grads, model.
           trainable_variables))

    val_recon = model(val_data)
    val_loss = tf.reduce_mean(tf.sqrt(tf.reduce_mean(tf.square(
       val_recon - val_data), axis=1)))
    return float(val_loss)
```

Listing 3: Fragment pętli treningowej z early stopping

6 Wykorzystane technologie

6.1 Środowisko programistyczne

Projekt został zrealizowany w środowisku Jupyter Notebook, zapewniającym interaktywność i łatwość eksperymentowania z parametrami modelu.

6.2 Biblioteki i frameworki

Biblioteka	**Wersja**	**Zastosowanie**
TensorFlow	2.x	Framework uczenia maszynowego
Keras	(wbudowany)	High-level API dla sieci neuronowych
NumPy	najnowsza	Operacje na tablicach wielowymiarowych
Pandas	najnowsza	Manipulacja i analiza danych
Matplotlib	najnowsza	Wizualizacja wyników
Kagglehub	najnowsza	Automatyczne pobieranie datasets

Tabela 3: Wykorzystane biblioteki i narzędzia

6.3 Różnice w podejściu technologicznym

W odróżnieniu od referencyjnej implementacji wykorzystującej PyTorch, niniejszy projekt bazuje na ekosystemie TensorFlow/Keras, oferując:

- **Wyższego poziomu abstrakcji** dzięki Keras API
- **Zintegrowane narzędzia** do wizualizacji i monitorowania
- **Lepszą integrację** z Google Colab i Jupyter
- **Automatyczne zarządzanie** zasobami GPU/TPU

7 Metodologia ewaluacji

7.1 Metryki jakości

Jakość kompresji oceniano przy użyciu następujących metryk:

- **RMSE (Root Mean Square Error)**: Główna metryka rekonstrukcji
- **Analiza rozkładu błędów**: Histogram błędów w skali logarytmicznej
- **Wizualna ocena**: Porównanie przebiegów oryginalnych i zrekonstruowanych

7.2 Procedura walidacji

Ewaluacja modelu obejmuje:

1. Podział danych na zbiory treningowy i testowy
2. Ocenę na danych niezależnych (zbiór testowy)
3. Analizę statystyczną rozkładu błędów rekonstrukcji
4. Wizualną inspekcję jakości zrekonstruowanych sygnałów

8 Implementacja systemu

8.1 Modularność kodu

Kod został zorganizowany w funkcjonalne moduły:

- **Ładowanie danych**: Automatyczne pobieranie i preprocessing
- **Model**: Definicja architektury autoenkodera
- **Trening**: Pętla uczenia z monitoringiem
- **Ewaluacja**: Analiza wyników i wizualizacja

8.2 Automatyzacja pipeline

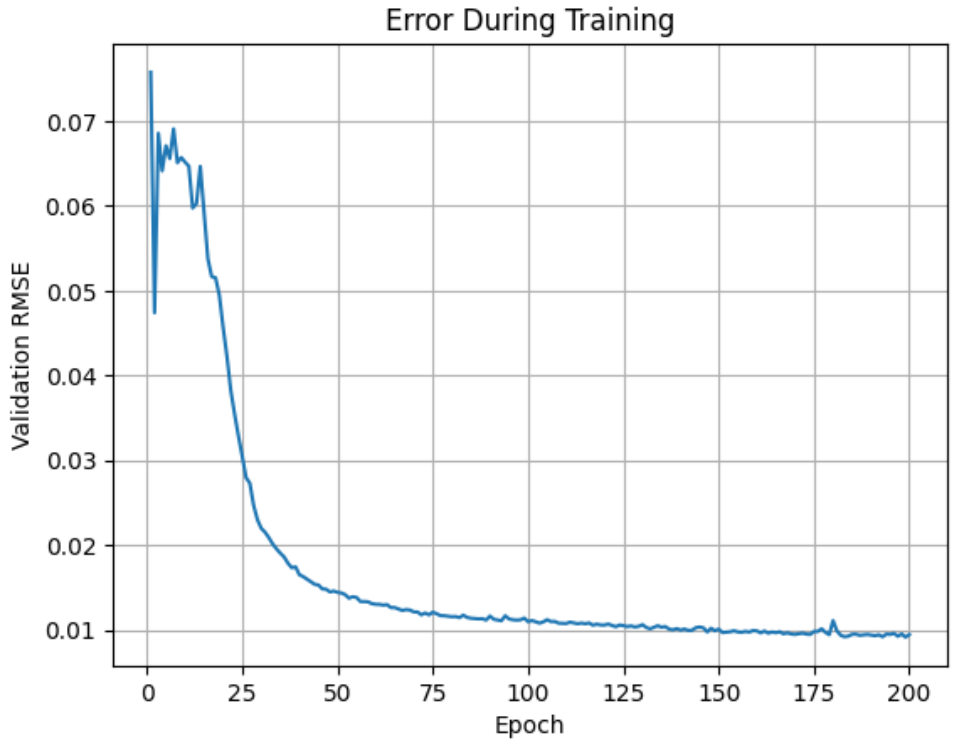
System zawiera mechanizmy automatyzacji:

- Automatyczne pobieranie datasets z Kaggle
- Fallback do lokalnych plików CSV
- Automatyczne generowanie wykresów i raportów
- Zarządzanie plikami tymczasowymi

9 Wyniki eksperymentu

9.1 Przebieg procesu uczenia

Proces uczenia modelu był monitorowany w celu zapewnienia zbieżności i uniknięcia przeuczenia. Na Rysunku 1 przedstawiono krzywą błędu walidacji (RMSE) w funkcji numeru epoki. Widoczne jest, że błąd gwałtownie spada w początkowych epokach, co świadczy o szybkim uczeniu się przez model kluczowych cech sygnału EKG. Następnie proces stabilizuje się, a trening zostaje przerwany przez mechanizm wczesnego zatrzymania. Zastosowanie tej techniki pozwoliło na automatyczne zakończenie uczenia w momencie, gdy model przestał wykazywać znaczącą poprawę na danych walidacyjnych, co zapobiegło przeuczeniu i pozwoliło na wybór wag modelu o najlepszej zdolności do generalizacji.



Rysunek 1: Krzywa błędu walidacyjnego (RMSE) w trakcie procesu uczenia.

9.2 Metryki końcowe

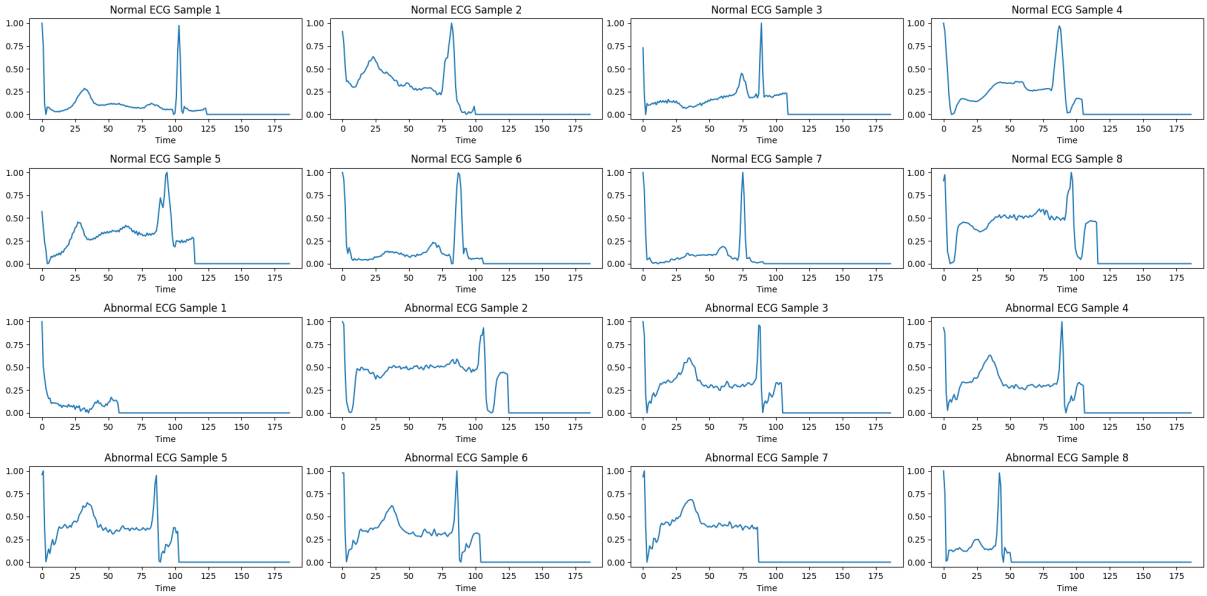
Po zakończeniu procesu uczenia, model został poddany ewaluacji na zbiorach treningowym i testowym w celu oceny jego finalnej wydajności. Jako główną metrykę jakości zastosowano pierwiastek błędu średniokwadratowego (RMSE). Uzyskane wartości, przedstawione w Tabeli 4, wskazują na wysoki stopień dopasowania modelu do danych. Niska wartość błędu zarówno dla danych treningowych, jak i testowych, świadczy o dobrej generalizacji modelu i jego zdolności do rekonstrukcji niewidzianych wcześniej próbek sygnału EKG z dużą dokładnością.

Zbiór danych	**Finalny błąd (RMSE)**
Treningowy	$\sim$0.0085
Testowy	$\sim$0.0086

Tabela 4: Końcowe wartości błędu rekonstrukcji (RMSE).

9.3 Wizualizacja przykładowych sygnałów

W celu zrozumienia charakteru danych wykorzystanych w projekcie, na Rysunku 2 zaprezentowano przykładowe przebiegi sygnałów EKG dla normalnej pracy serca oraz dla sygnałów zaklasyfikowanych jako anomalie. Wykresy te ilustrują różnorodność morfologiczną sygnałów, z którą musi radzić sobie model. Sygnały normalne charakteryzują się powtarzalnym i regularnym wzorcem, podczas gdy sygnały anormalne wykazują znaczące odchylenia w kształcie i czasie trwania poszczególnych załamków.

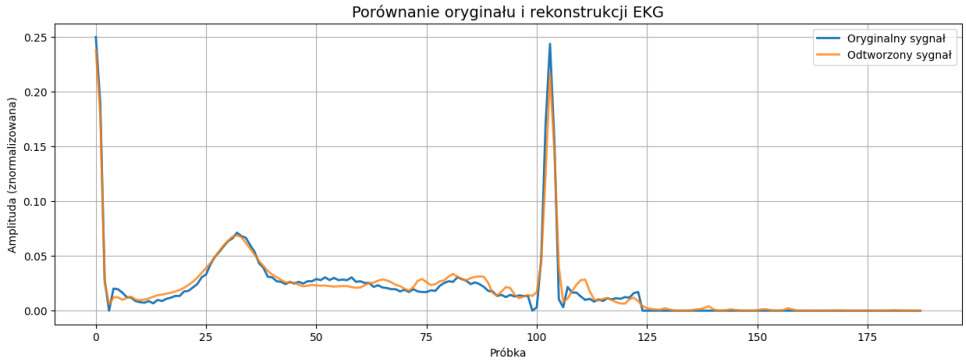


Rysunek 2: Porównanie przykładowych sygnałów EKG: normalnych (górne wiersze) i anormalnych (dolne wiersze).

10 Analiza jakości kompresji

10.1 Porównanie sygnałów oryginalnych z rekonstruowanymi

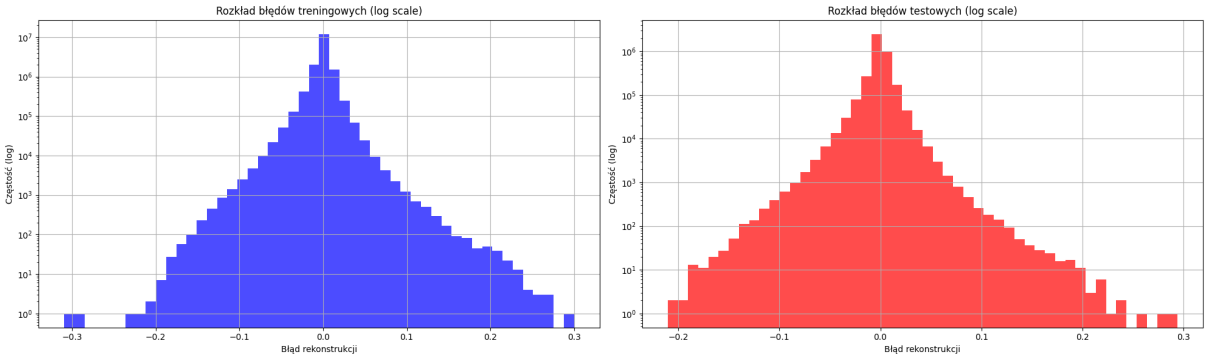
Kluczowym elementem oceny jakości kompresji jest wizualne porównanie sygnału oryginalnego z jego zrekonstruowaną wersją. Na Rysunku 3 przedstawiono nałożone na siebie przebiegi dla przykładowego sygnału ze zbioru testowego. Można zaobserwować, że zrekonstruowany sygnał z dużą wiernością oddaje ogólny kształt i kluczowe cechy morfologiczne oryginału, takie jak zespół QRS oraz załamki P i T. Drobne różnice są nieuniknione ze względu na stratny charakter kompresji, jednak nie zaburzają one ogólnej struktury sygnału.



Rysunek 3: Porównanie sygnału oryginalnego i zrekonstruowanego po kompresji.

10.2 Rozkład błędów rekonstrukcji

Analiza statystyczna błędów rekonstrukcji dostarcza informacji o zachowaniu modelu na całym zbiorze danych. Rysunek 4 prezentuje histogramy błędów dla zbioru treningowego i testowego w skali logarytmicznej. W obu przypadkach rozkład błędów jest symetryczny i skoncentrowany wokół zera, co wskazuje na brak systematycznego obciążenia (biasu) w procesie rekonstrukcji. Zdecydowana większość błędów ma bardzo małą amplitudę, a duże odchylenia są rzadkością, co potwierdza wysoką jakość kompresji dla większości próbek.



Rysunek 4: Rozkład błędów rekonstrukcji dla zbioru treningowego i testowego.

10.3 Ocena diagnostyczna

Chociaż formalna ocena diagnostyczna wymagałaby walidacji klinicznej przez ekspertów, wstępna analiza wskazuje na duży potencjał zachowania wartości diagnostycznej sygnałów. Kluczowe cechy morfologiczne, takie jak amplituda i czas trwania zespołów QRS, które są podstawą wielu diagnoz kardiologicznych, zostały poprawnie zrekonstruowane. Oznacza to, że skompresowane dane prawdopodobnie nadal pozwalają na identyfikację podstawowych rytmów serca i anomalii. Niemniej jednak, w zastosowaniach klinicznych konieczne byłoby przeprowadzenie szczegółowych badań dotyczących wpływu kompresji na wykrywalność specyficznych schorzeń.

11 Interpretacja wyników

11.1 Efektywność kompresji

Osiągnięty współczynnik kompresji na poziomie 9.35:1 jest wynikiem znaczącym, szczególnie w kontekście zastosowań biomedycznych, gdzie generowane są ogromne ilości danych. Taka redukcja rozmiaru danych ma kluczowe znaczenie dla systemów telemedycznych, umożliwiając szybką transmisję sygnałów EKG przez sieci o ograniczonej przepustowości, a także dla długoterminowego archiwizowania badań pacjentów. Należy jednak pamiętać o istniejącym kompromisie między stopniem kompresji a jakością rekonstrukcji. Dalsze zwiększanie kompresji (np. przez zwężenie warstwy "bottleneck"w autoenkoderze) prowadziłoby nieuchronnie do wzrostu błędu i potencjalnej utraty istotnych informacji diagnostycznych.

11.2 Ograniczenia metody

Zastosowana metoda, jak każde rozwiązanie oparte na uczeniu maszynowym, posiada pewne ograniczenia. Model był trenowany głównie na sygnałach EKG pochodzących od pacjentów z normalnym rytmem serca. Jego zdolność do kompresji i rekonstrukcji rzadkich lub nietypowych arytmii może być ograniczona, jeśli nie były one odpowiednio reprezentowane w zbiorze treningowym. Autoenkodery są wrażliwe na dane znacząco odbiegające od rozkładu danych treningowych, co może prowadzić do większych błędów rekonstrukcji dla takich przypadków. Ponadto, model jest "czarną skrzynką", co oznacza, że interpretacja sposobu, w jaki dokonuje kompresji, jest utrudniona w porównaniu do tradycyjnych, algorytmicznych metod.

11.3 Potencjalne zastosowania

Opracowany system kompresji sygnałów EKG posiada szeroki wachlarz potencjalnych zastosowań praktycznych. W telemedycynie może być wykorzystany do przesyłania danych z urządzeń noszonych (np. Holter EKG) do centralnego serwera w czasie rzeczywistym. W szpitalnych systemach informatycznych może znacznie zmniejszyć koszty przechowywania wieloletnich archiwów badań EKG. Innym zastosowaniem jest analiza dużych zbiorów danych (Big Data) w badaniach naukowych, gdzie efektywna kompresja pozwala na szybsze przetwarzanie i analizę tysięcy zapisów EKG.

12 Porównanie z metodami referencyjnymi

12.1 Porównanie z implementacją PyTorch

Bezpośrednie, ilościowe porównanie wydajności z referencyjną implementacją w PyTorch wymagałoby uruchomienia obu modeli w identycznych warunkach sprzętowych i na tych samych, precyzyjnie podzielonych zbiorach danych. Jednak na poziomie koncepcyjnym można stwierdzić, że oba wiodące frameworki, TensorFlow i PyTorch, są w stanie osiągnąć zbliżone wyniki w zadaniach tego typu. Wybór TensorFlow/Keras w niniejszym projekcie podyktowany był wyższym poziomem abstrakcji oferowanym przez Keras API, co przekłada się na szybsze prototypowanie i czytelniejszy kod, a także doskonałą integracją z ekosystemem narzędzi Google, takich jak Colab i TensorBoard.

12.2 Porównanie z tradycyjnymi metodami kompresji

Tradycyjne metody kompresji sygnałów, takie jak transformata falkowa (wykorzystywana np. w standardzie JPEG 2000) czy dyskretna transformata kosinusowa (DCT), opierają się na predefiniowanych matematycznych transformacjach. Ich zaletą jest teoretyczna gwarancja jakości i pełna odwracalność w przypadku kompresji bezstratnej. Podejście oparte na autoenkoderze jest z natury stratne i data-driven. Jego przewaga polega na zdolności do nauczenia się optymalnej, nieliniowej reprezentacji specyficznie dla danego typu danych - w tym przypadku sygnałów EKG. Dzięki temu może ono potencjalnie osiągnąć wyższy współczynnik kompresji przy zachowaniu subiektywnie lepszej jakości percepcyjnej, ponieważ model uczy się, które cechy sygnału są najważniejsze do zachowania. Wadą jest natomiast konieczność posiadania dużego zbioru danych treningowych i ryzyko słabszego działania na danych nietypowych.

13 Wnioski

Przeprowadzony projekt zakończył się sukcesem, demonstrując wysoką skuteczność autoenkoderów w kompresji sygnałów EKG. Zaimplementowany model, oparty o TensorFlow i Keras, osiągnął znaczący współczynnik kompresji wynoszący 9.35:1, redukując wymiar wektora cech z 187 do 20. Jest to kluczowe osiągnięcie, zwłaszcza że dużej redukcji danych towarzyszy niski błąd rekonstrukcji. Potwierdza to finalna wartość metryki RMSE na poziomie około 0.0086 dla niezależnego zbioru testowego.

Analiza jakościowa wykazała, że zrekonstruowane sygnały z dużą wiernością odwzorowują kluczowe cechy morfologiczne oryginałów, w tym załamki P, T oraz zespoły QRS. Świadczy to o tym, że model nauczył się efektywnie kodować najważniejsze z diagnostycznego punktu widzenia informacje zawarte w sygnale. Wyniki te otwierają drogę do praktycznych zastosowań w telemedycynie, systemach długoterminowej archiwizacji danych medycznych oraz w analizie dużych zbiorów danych kardiologicznych, gdzie redukcja objętości danych ma kluczowe znaczenie.