**Rekomendowana Struktura Kodu Notatnika**

**Część 1: Setup i Konfiguracja (Górna część notatnika)**

Tutaj przygotowujemy wszystko, co będzie potrzebne, i definiujemy wszystkie "pokrętła", którymi będziemy sterować.

1. **Importy bibliotek**
   * import tensorflow as tf, import numpy as np, import matplotlib.pyplot as plt, itd.
   * Zawsze na samej górze.
2. **Konfiguracja Globalna (CONFIG)**
   * To **bardzo ważny krok!** W jednym miejscu zdefiniuj wszystkie kluczowe parametry. Dzięki temu, gdy zechcesz coś zmienić (np. rozmiar obrazu), zrobisz to w jednym miejscu, a nie w dziesięciu.
   * IMAGE\_SIZE = (224, 224)
   * BATCH\_SIZE = 32
   * EPOCHS = 50
   * LEARNING\_RATE = 0.0001
   * BASE\_MODEL\_NAME = 'google/vit-base-patch16-224-in21k'
   * ROOT\_DATA\_DIR = Path('/content/data/grouped')
3. **Inicjalizacja Strategii (TPU/GPU)**
   * Kod do wykrywania i inicjalizacji TPU lub GPU (strategy = ...).
   * Powinien być na początku, zaraz po konfiguracji.
4. **Zdefiniowanie Funkcji Pomocniczych**
   * Zgrupuj w jednym miejscu **definicje** wszystkich funkcji, których będziesz używać później. To sprawia, że główna część kodu jest bardzo czysta.
   * def load\_and\_preprocess\_image(path): ...
   * def create\_model(config, num\_classes): ...
   * def plot\_and\_save\_history(history, filename): ...

**Część 2: Przygotowanie Danych (Data Pipeline)**

Teraz, gdy mamy już narzędzia (funkcje) i konfigurację, zajmujemy się danymi.

1. **Wczytanie Ścieżek do Plików i Etykiet**
   * Pętla, która przechodzi przez foldery (np. używając pathlib.glob) i tworzy listy file\_paths i labels.
2. **Podział Danych na Zbiory (Train, Validation, Test)**
   * Użycie train\_test\_split do podziału list file\_paths i labels.
   * Dzielimy tylko ścieżki, a nie wczytane obrazy – to oszczędza pamięć.
3. **Obliczenie Wag Klas (Class Weights)**
   * Jeśli masz niezbalansowany zbiór, to jest idealne miejsce, aby na podstawie etykiet treningowych (y\_train) obliczyć wagi klas.
4. **Stworzenie Potoków Danych tf.data.Dataset**
   * train\_dataset = tf.data.Dataset.from\_tensor\_slices(...)
   * validation\_dataset = ...
   * Użycie metody .map() z funkcją load\_and\_preprocess\_image zdefiniowaną w kroku 4.
   * Użycie .shuffle(), .batch(), .prefetch().

**Część 3: Budowa i Trening Modelu (Główna akcja)**

Mamy dane, więc teraz budujemy i trenujemy model.

1. **Stworzenie i Kompilacja Modelu**
   * Wywołaj funkcję zdefiniowaną w kroku 4, aby stworzyć instancję modelu: model = create\_model(...).
   * To wszystko powinno dziać się wewnątrz with strategy.scope():.
   * Kompilacja modelu: model.compile(...).
   * Wyświetlenie podsumowania: model.summary().
2. **Trening Modelu**
   * Najważniejsza linijka: history = model.fit(...).
   * Tutaj podajesz train\_dataset, validation\_dataset, epochs, class\_weight itd.

**Część 4: Ewaluacja i Zapisanie Wyników**

Trening skończony, czas na analizę i zapisanie efektów naszej pracy.

1. **Analiza i Wizualizacja Historii Treningu**
   * Wywołanie funkcji z kroku 4: plot\_and\_save\_history(history, 'wyniki.png').
2. **Ewaluacja na Zbiorze Testowym**
   * results = model.evaluate(test\_dataset).
   * print(results)
3. **(Opcjonalnie) Bardziej zaawansowana analiza**
   * Generowanie macierzy pomyłek (confusion matrix), raportu klasyfikacji.
4. **Zapisanie Wytrenowanego Modelu**
   * Ostatni krok: model.save('retinopathy\_classifier\_v1.h5').

**Podsumowując, Twoja kolejność była dobra, ale kluczowe zmiany to:**

* **CONFIG** na sam szczyt.
* **FUNKCJE POMOCNICZE** zdefiniowane razem w jednym bloku, przed ich użyciem.
* Wyraźne rozróżnienie między **definicją funkcji** (w części 1) a jej **wywołaniem** (w częściach 3 i 4).

Taka struktura sprawi, że Twój notatnik będzie jak dobrze napisana instrukcja: najpierw narzędzia i składniki, potem przepis krok po kroku.